Limited Disclosure Version

Vol.2

FE Exam Preparation Book

Preparation Book for Fundamental Information Technology Engineer Examination

Preparation for Afternoon Exam

IPA

Information-Technology Promotion Agency, Japan

FE Exam Preparation Book Vol. 2

Table of Contents

PREPARATION FOR AFTERNOON EXAM

PREPARATION FOR
AFTERNOON EXAM

The Afternoon Exam questions are formulated from the following eight fields: Hardware, Software, Data Structure and Database, Communication Network, Information Processing Technology, Algorithms, Program Design, and Program Development. Here, the pairs of questions and answers in each field are provided. All questions are used in the past exams.

1 Hardware

[Scope of Questions]
Numeric representation,
Character representation,
Image/sound representation, Processor,
Memory, I/O device,
Execution of operation,
Addressing scheme,
I/O process execution,
System configuration, etc.

Question 1

Q1. Read the following description about the execution of an instruction, and then answer Subquestion.

There is a computer with a main memory capacity of 65,536 words, wherein one word consists of 16 bits. It has four general purpose registers (0 through 3) and a program register ("PR" below). The format of an instruction (2 words in length) is as follows:

OP	R	X	I	D	adr

OP: Specifies an operation code in 8 bits. In this example, the following three operation codes are used.

- 20_{16}: Loads the effective address into the general purpose register specified by R.
- 21_{16}: Loads the contents of the word indicated by the effective address into the general purpose register specified by R.
- FF_{16}: Ends execution.

R: Specifies a general purpose register number (0 through 3) in two bits.

X: Specifies an index register number (1 through 3) in two bits. The general purpose register at the specified number is used as an index register. However, if "0" is specified, no index register-based modification is made.

I: Specifies "1" in a single bit for indirect addressing; otherwise, specifies "0".

D: Three extension bits which are always "0".

adr: Specifies an address in 16 bits.

The effective address of instruction is calculated as shown in the following table.

Table Relationships Between X, I, and Effective Addresses

X	I	Effective address
0	0	adr
1 through 3	0	adr + (X)
0	1	(adr)
1 through 3	1	(adr + (X))

Note ( ): The parentheses are used to denote the content stored in the register or address inside the parentheses.

Subquestion

From the answer group below, select the correct answers to be inserted in the blanks [ ] in the following description.

When the contents of the general purpose registers and the main memory had the values shown in the figure below (values are in HEX notation), 0100_{16} was set in PR and the program was executed. In this example, the instruction at the address 0100_{16} loads the contents 0113_{16} at the address $011B_{16}$ (underlined) to general purpose register 0.

After the execution ends, [A] is set to general purpose register 0, [B] to general purpose register 1, [C] to general purpose register 2, and [D] to general purpose register 3.

General purpose register 0: 0003 1: 0000 2: 0000 3: 0000

Program register PR: 0100

Main memory

Address	+ 0	+ 1	+ 2	+ 3	+ 4	+ 5	+ 6	+ 7
00F8	0000	0000	0000	0000	0000	0000	0000	0000
0100	2100	011B	20C0	0003	2170	0111	21B8	011A
0108	FF00	0000	0000	0000	0000	0000	0000	0000
0110	0000	0001	0002	0003	0004	0005	0006	0007
0118	0110	0111	0112	0113	0114	0115	0116	0117
0120	0118	0119	011A	011B	011C	011D	011E	011F

Fig. Values in General Purpose Registers, PR, and Main Memory

Answer group:

a) 0000_{16} b) 0001_{16} c) 0002_{16}

d) 0003_{16} e) 0004_{16} f) 0005_{16}

g) 0006_{16} h) 0113_{16} i) 0115_{16}

Answer 1

Trace of General-Purpose Register Values in Instruction Execution

Correct Answer

A – h, B – e, C – f, D – d

Comments

Today, it is rare to program in assembly language, so many people understandably find it difficult to answer questions about machine language. Also, many examinees seemed to have no clue as to how to read the diagram showing the contents of the main memory (called a dump list).

However, the correct answer can be obtained if you understand the operation principle of the CPU, i.e., repeatedly fetching an instruction and executing it, and, in this case, fetching one instruction of two-word (= 32 bits) length at a time and executing it, beginning at the address of the program register (program counter). This question might give an advantage to those who have knowledge of the programming language CASL, but the question and examples are given in detail, so you can score points by carefully tracing the program, paying attention to the idea of the indirect addressing.

Two addressing modes — indexed addressing and indirect addressing — are used here. Indexed addressing involves the calculation of the effective address by adding the contents of the index register to the designated address value. Indirect addressing involves taking the value stored in the main memory designated by the address part of the instruction as the effective address.

The format of an instruction is as shown below; you must pay close attention to the parts R, X, and I below when tracing the program:

[Format of an instruction (2-word length)]

OP	R	X	I	D	adr
8 bits	2	2	1	3	16 bits

- R (0 – 3 : general-purpose register number)
- X (1 – 3 : index register number, 0 : not modified by index register)
- I (1 : indirect addressing applied , 0 : indirect addressing not applied)

First, the beginning address of the program is 0100_{16}, which is set in the program register (PR), so the program instructions are stored, beginning at this address. If we label each 2-word-long instruction in the contents of the main memory, we get the following:

[Contents of the main memory] Values as hexadecimal numbers

Address	+0	+1	+2	+3	+4	+5	+6	+7
00F8	(1) 0000	0000	(2) 0000	0000	(3) 0000	0000	(4) 0000	0000
0100	2100	011B	20C0	0003	2170	0111	21B8	011A
0108	(5) FF00	0000	0000	0000	0000	0000	0000	0000
0110	0000	0001	0002	0003	0004	0005	0006	0007
0118	0110	0111	0112	0113	0114	0115	0116	0117
0120	0118	0119	011A	011B	011C	011D	011E	011F

Let us now examine the contents of these instructions as we trace the program in order. To calculate effective addresses, we need to refer to the table given in the question (relation among X, I, and the effective address) and calculate carefully.

(1) 2100 011B

- This is the first instruction in the program.
- Bit expression of the underlined 00: 0000 0000 → R : 00 (general-purpose register 0), X : 00, I : 0.
- Instruction code 21_{16}→ The content $(0113)_{16}$ of the word designated by effective address $(011B)_{16}$ is loaded into general-purpose register 0, which is designated by R.
- General-purpose register 0 0003_{16} → 0113_{16}

(2) 20C0 0003

- Bit expression of the underlined C0: 1100 0000 → R: 11 (general-purpose register 3), X : 00, I : 0.
- Instruction code 20_{16} → Effective address (0003_{16}) is loaded into general-purpose register 3, which is designated by R.
- General-purpose register 3 0000_{16} → 0003_{16}

(3) 2170 0111

- Bit expression of the underlined 70: 0111 0000 →R : 01(general-purpose register 1), X : 11, I : 0.
- Since X:11 (referring to general-purpose register 3), address modification occurs, using general-purpose register 3 as the index register.
- Calculation of the effective address
- adr + (X) = 0111_{16} + (general-purpose register 3) = 0111_{16} + 0003_{16} = 0114_{16}

  ↓

  Content stored in general-purpose register 3
- Instruction code 21_{16} → The content $(0004)_{16}$ of the word designated by effective address $(0114)_{16}$ is loaded into general-purpose register 1 designated by R.
- General-purpose register 1 0000_{16} → 0004_{16}

(4) 21B8 011A

- Expression of the underlined B8: 1011 1000 → R : 10 (general-purpose register 2), X : 11, I : 1.
- Since X:11 (referring to general-purpose register 3), address modification occurs, using general-purpose register 3 as the index register. Also, since I:1, indirect addressing also takes place.
- Calculation of the effective address

(adr + (X)) = ($011A_{16}$ + (general-purpose register 3)) content stored in general-purpose register 3
= ($011A_{16}$ + 0003_{16})
= ($011D_{16}$) ……content stored in $011D_{16}$
= 0115_{16}

- Instruction code 21_{16} → The content $(0005)_{16}$ of the word designated by effective address $(0115)_{16}$ is loaded into general-purpose register 2, which is designated by R.
- General-purpose register 2 0000_{16} → 0005_{16}

(5) FF00 0000

- Instruction code FF_{16} → Execution of the program ends.

As seen above, **it is essential that you trace the results of executing instructions carefully; by doing so, you can find the correct answers**. The following table summarized the changes in register values due to instruction execution:

Instruction	A	B	C	D
	GR0	GR1	GR2	GR3
(Beginning)	0003	0000	0000	0000
(1) 2100 011B	0113	↓	↓	↓
(2) 20C0 0003	↓	↓	↓	0003
(3) 2170 0111	↓	0004	↓	↓
(4) 21B8 011A	↓	↓	0005	↓
(5) FF00 0000	↓	↓	↓	↓
(Answers)	h	e	f	d

2 Software

[Scope of Questions]
System software, Application software,
Software package, OS functions,
Programming language,
Language processor, Program execution, etc.

Question 1

Q1. Read the following description about a scheduler, and then answer Subquestion.

[Description of Scheduler]

(1) The scheduler is implemented on a single processor using multiple programming.

(2) A time slice is 50 milliseconds.

(3) There is no priority among processes, and round robin scheduling is used. .

(4) If I/O occurs while a process is running, the next process is executed. Also, when the turn comes to a process waiting for I/O, that process is skipped, and the next process is executed.

Subquestion

From the answer group below, select the correct answers to be inserted in the blanks ☐ in the following description.

There are three processes: P1, P2, and P3. The figure shows the sequence in which the processor and I/O are used, and the amount of time during which they are used, when each process is executed independently.

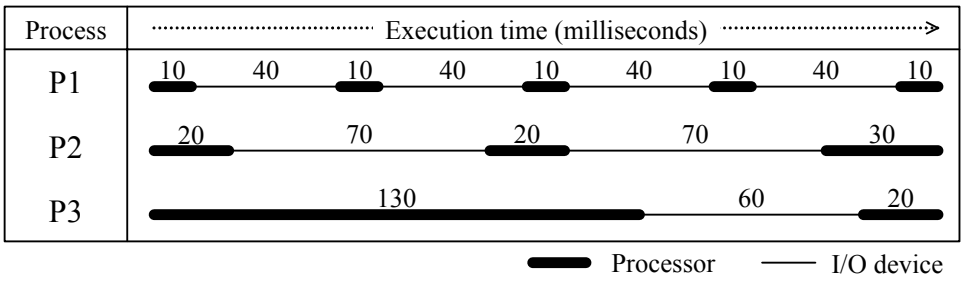


Fig. Sequence and Time Length of Use of the Processor and I/O Devices

Now, assume that the three processes P1, P2, and P3 are executed according to round robin scheduling in the sequence P1 → P2 → P3 → P1 → ... Overhead caused by switching between processes can be ignored.

(1) When the three processes use different I/O devices α, β and γ respectively, the process that finishes first is ☐A and the time from the start of P1 to the end of all the processes is ☐B milliseconds.

(2) When process P1 uses I/O device α and processes P2 and P3 both use I/O device β (one process has to wait until the other that starts to use the I/O device earlier finishes), the process(es) ☐C take(s) a longer time than the case of (1) and the extended duration is ☐D milliseconds .

Answer group for A and C:

a) P1	b) P1 and P2	c) P1 and P3
d) P2	e) P2 and P3	f) P3

Answer group for B:

a) 250	b) 260	c) 270	d) 280
e) 290	f) 300	g) 310	h) 320

Answer group for D:

a) 10	b) 20	c) 30	d) 40
e) 50	f) 60	g) 70	h) 80

Answer 1

Scheduler

Correct Answer

A – d, B – f, C – f, D – c

Comments

This question involves a scheduler implementing multiprogramming. It is given that the round-robin scheduling is used, which switches the process after a certain period of time (called a time quantum or time slice) to allocate the execution privilege of the CPU. If the process is still running at the end of the quantum, the CPU is preempted and given to another process. Here are some important points on the scheduler:

(1) It uses a single processor.
(2) The time slice is 50 milliseconds, i.e., the maximum usage time of the processor is 50 milliseconds.
(3) There is no priority among processes, and the round-robin scheduling is used.
(4) If I/O occurs while a process is running, the next process is executed.
(5) If a turn comes to a process waiting for I/O, that process is skipped, and the next process is executed.

The key point for answering this question is to understand the contents of the scheduler and to create a time chart correctly and quickly. One important point to consider is this: if the turn comes to a process waiting for I/O and thus the process is not executed, it is skipped; however, even when that process gets back to an executable state, it is not executed immediately at that point. The process is always executed in a certain order. This must be kept in mind as the time chart is prepared.

[Blanks A and B]

The execution order of the three processes P1, P2, and P3 is P1 → P2 → P3 → P1 →…Based on the order and the time of the processor and the I/O given in the diagram in the question, one can create a time chart as shown below.

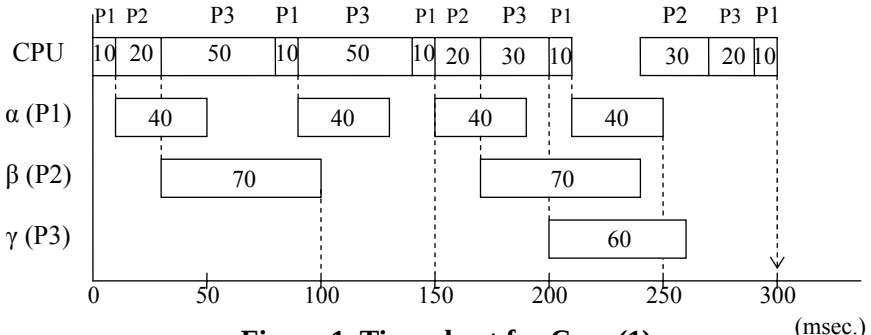


Figure 1. Time chart for Case (1)

Hence, the processes finish in the order of P2 (270) → P3 (290) → P1 (300), where the numbers in the parentheses indicate the time taken to finish. Thus, the first process to finish is P2 (answer (d)), and it takes 300 milliseconds (answer (f)) to finish all processes.

[Blanks C and D]

The time chart below is for the case in which P1 uses the I/O device α while P2 and P3 use the same I/O device β.

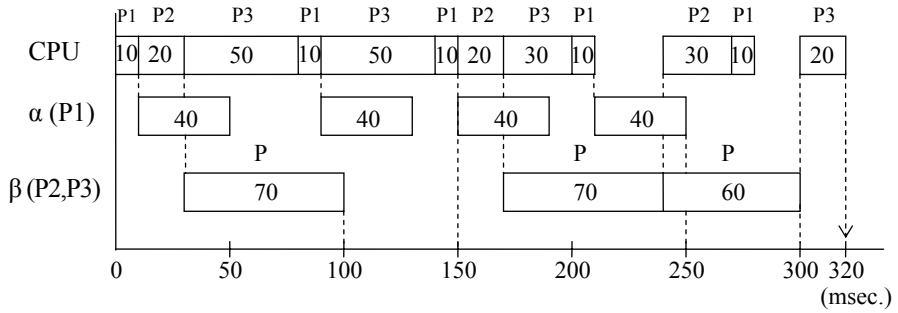


Figure 2. Time chart for Case (2)

From Figure 2 we can see that the processes finish in the order of P2 (270) → P1 (280) → P3 (320). Compared to Case (1), it is Process P3 that takes longer to finish, and the extended duration of that process (P3) is 320 – 290 = 30 milliseconds. Hence, the correct answer for C is (f), and the correct answer for D is (c).

However, one may interpret the phrase "the extended duration" in the question to mean "the whole duration compared to the case of (1)". In that case, the answer would be 20 milliseconds. This would change the correct answer for D to "20 milliseconds (b)". There is, therefore, some ambiguity in the question text, but here we take the term "the extended duration" to mean "the extended duration for process P3".

3 Data Structure and Database

[Scope of Questions]
Fundamental data structure,
Types and features of storage media,
File organization methods,
Types and characteristics of database,
Database language,
Data manipulation using SQL, etc.

Question 1

Q1. Read the following description about a relational database, and then answer Subquestions 1 and 2.

There are two tables named PRODUCT_TABLE and WHOLESALER_TABLE illustrated below. The name of each field is shown at the top. The underlined field indicates the primary key.

PRODUCT_TABLE

PRODUCT_CODE	PRODUCT_NAME	UNIT_PRICE	WHOLESALER_CODE
S0001	Pencil	10	T0306
S0002	Eraser	50	T1020
⋮			

WHOLESALER_TABLE

WHOLESALER_CODE	WHOLESALER_NAME	PHONE_NO	REMARKS
T0001	Smith Company	03-1234-5678	
T0002	Regan Company	03-8765-4321	
⋮			

Subquestion 1

From the answer group below, select the correct answers to be inserted in the blanks [ ] in the SQL statement that defines the following ORDER_ TABLE.

ORDER_TABLE

ORDER_NO	ORDER_DATE	PRODUCT_CODE	QUANTITY
3001	2007-03-01	S0110	1000
3002	2007-03-02	S0054	2000
⋮			

```
CREATE TABLE ORDER_TABLE
    (ORDER_NO NUMERIC(4) NOT NULL,
    ORDER_DATE DATE NOT NULL,
    PRODUCT_CODE CHAR(5) NOT NULL,
    QUANTITY NUMERIC(5) NOT NULL,
    [ A ] KEY (ORDER_NO),
    [ B ] KEY (PRODUCT_CODE) [ C ] PRODUCT_TABLE)
```

Answer group:

a) FOREIGN b) INDEX c) MAIN
d) PRIMARY e) REFERENCES f) UNIQUE
g) USING

Subquestion 2

From the answer group below, select the correct answers to be inserted in the blanks [ ] in the SQL statement that defines order slip view, which has the same number of lines as the order table so that it can be used to confirm the contents of the order.

ORDER_SLIP

WHOLESALER_NAME	ORDER_DATE	PRODUCT_NAME	QUANTITY	AMOUNT
Thomson Company	2007-03-01	Notebook	100	10000
Holland Company	2007-03-02	Stick glue	200	20000
⋮				

```
CREATE VIEW ORDER_SLIP
  AS SELECT W.WHOLESALER_NAME, O.ORDER_DATE,
            P.PRODUCT_NAME, O.QUANTITY, [ D ]
  FROM ORDER_TABLE O, PRODUCT_TABLE P, WHOLESALER_TABLE W [ E ]
```

Answer group for D:

a) `P.UNIT_PRICE * O.QUANTITY AS AMOUNT`
b) `SUM(P.UNIT_PRICE * O.QUANTITY) AS AMOUNT`
c) `AMOUNT IS P.UNIT_PRICE * O.QUANTITY`
d) `AMOUNT IS SUM(P.UNIT_PRICE * O.QUANTITY)`

Answer group for E:

a)
```
IN O.PRODUCT_CODE = P.PRODUCT_CODE
 AND P.WHOLESALER_CODE = W.WHOLESALER_CODE
```
b)
```
IN O.PRODUCT_CODE = P.PRODUCT_CODE
 OR P.WHOLESALER_CODE = W.WHOLESALER_CODE
```
c)
```
WHERE O.PRODUCT_CODE = P.PRODUCT_CODE
 AND P.WHOLESALER_CODE = W.WHOLESALER_CODE
```
d)
```
WHERE O.PRODUCT_CODE = P.PRODUCT_CODE
 OR P.WHOLESALER_CODE = W.WHOLESALER_CODE
```

Answer 1

SQL Sentences to Define Tables

Correct Answer

[Subquestion 1] A – d, B – a, C – e

[Subquestion 2] D – a, E – c

Comments

[Subquestion 1]

If you have never seen a `CREATE TABLE` statement, you may be dismayed at first, but you can probably figure the question out to a certain extent if you review it with composure. Since the order number in the order table is underlined, ORDER_NO is the primary key. Therefore, it is reasonably assumed that blank A has something to do with the definition of the primary key, so the answer is (d) `PRIMARY`. You might think that the answer could be (f) `UNIQUE`, but there is no such designation as a unique key. It is possible to designate the term `UNIQUE` next to order number, but this designation is rarely seen in SQL today. Since Blank B designates the product code, if you noticed that PRODUCT_CODE is the primary key in PRODUCT_TABLE, you can identify that this designates a reference constraint of product code. For a reference constraint, i.e., definition of an external key, the correct answer is (a) `FOREIGN`. For blank C, since the primary key of the product table is referenced, the correct answer is (e) `REFERENCES`.

Note that ORDER_NO, etc., are followed by `NOT NULL`. This means that the value for the order number, etc., cannot be `NULL`.

[Subquestion 2]

A view is a virtual table temporarily created when referencing the view. The order slip view has fields including the wholesaler name from the wholesaler table, order date from the order table, and product name from the product table, so it cannot be defined unless the three tables are joined. The `FROM` phrase specifically mentions these three tables, so you just have to use a `WHERE` phrase to define the joining conditions. Blank E is that designation of the joining conditions, so the correct answer is (c) with an `AND` condition. Blank D corresponds to the amount in the order view, so the correct answer is (a) unit price * quantity = amount. This answer (a) has `AS` AMOUNT, which defines the field of the view as "AMOUNT". Unless the name of the field is designated with `AS`, the name cannot be determined by standard SQL. In Oracle, a commercial RDBMS, the calculation formula itself becomes the name in the absence of a specified name; this is a property unique to Oracle.

4 Communication Network

[Scope of Questions]
Data transmission,
Transmission control, TCP/IP, LAN, WAN,
Internet, Email, Web, etc.

Question 1

Q1. Read the following description about a communication network and then answer Subquestions 1, 2 and 3.

Company *A* plans to provide internal information service by installing an information-providing server which the head office and the factory can commonly use.

As shown in Figure 1, the LAN in the head office and that in the factory are connected via routers to the backbone network of Company *A*, and the backbone network is connected via a firewall to the Internet. TCP and IP are used at the transport layer and the network layer, respectively.

At the head office and the factory, each client uses a displaying program called a browser. The browser allows the client to communicate with a myriad of servers on the Internet with an application-layer protocol called HTTP. All computers on the network are set up in such a way that communication with the outside via routers always passes through proxy servers.

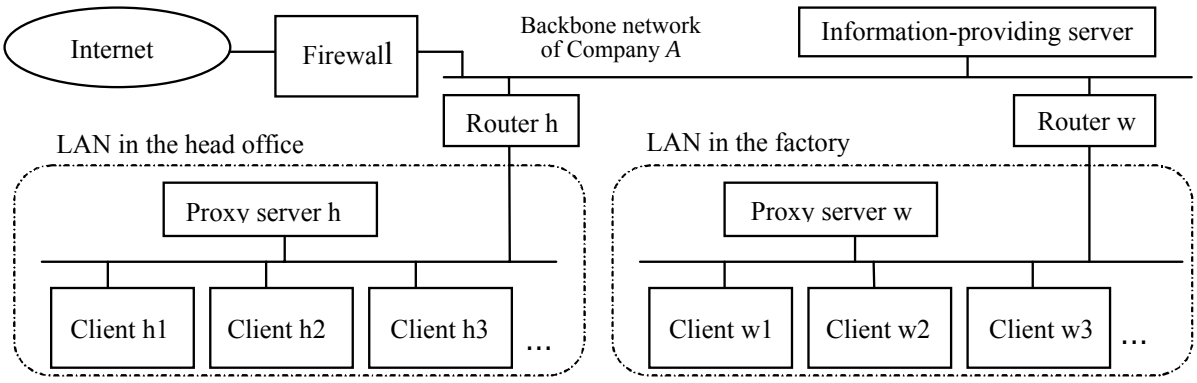


Fig. 1 Internal network of Company *A*

Figure 2 shows the communication path for the internal information-providing service. If HTTP, TCP and IP are used, the application layer directly receives the service from the transport layer.

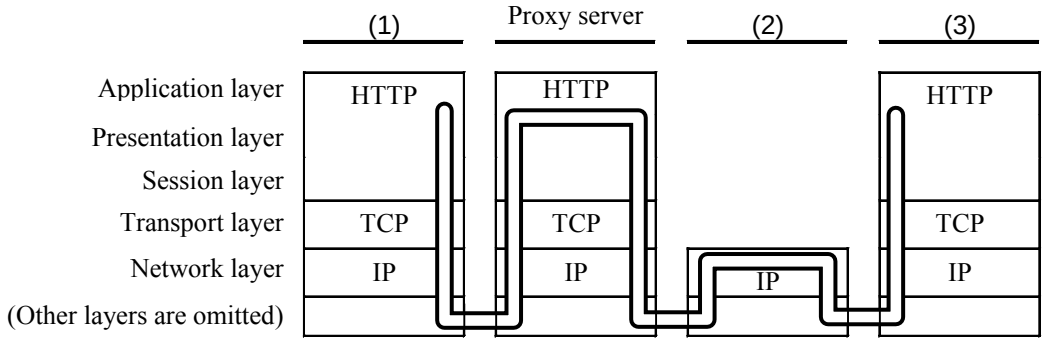


Fig. 2 Communication path for the internal information-providing service

Subquestion 1

In Figure 1, the client h1 must be authorized to pass (filter) through the router h if it communicates with the information-providing server. To be authorized, a correct source-to-destination setting must be established. From the answer group below, select the two correct source-to-destination settings.

Answer group

	Source	Destination
a)	Client h1	Information-providing server
b)	Client h1	Proxy server h
c)	Client h1	Firewall
d)	Information-providing server	Client h1
e)	Information-providing server	Proxy server h
f)	Proxy server h	Client h1
g)	Proxy server h	Information-providing server
h)	Proxy server h	Firewall
i)	Firewall	Client h1
j)	Firewall	Proxy server h

Subquestion 2

In Figure 2, a certain combination of different equipments is used in the computer network. From the answer group below, select the one correct combination that is used in Figure 2.

Answer group

	(1)	(2)	(3)
a)	Client	Information-providing server	Router
b)	Client	Router	Information-providing server
c)	Information-providing server	Client	Router
d)	Information-providing server	Router	Client
e)	Router	Client	Information-providing server
f)	Router	Information-providing server	Client

Subquestion 3

A problem occurred as described in the following. From the answer group below, select the possible cause of the problem.

All clients connecting to the head office LAN and to the factory LAN were able to communicate with the information-providing server and the servers on the Internet.

One client was added to the LAN at the factory. This new client was able to communicate directly with the other clients on the factory LAN but was unable to communicate with the information-providing server or the servers on the Internet.

All clients including the newly added client on the factory LAN used the same equipment, the same software and the same browser setting.

Answer group

a) A wrong IP address was set on the information-providing server, thus causing incorrect access control.
b) A wrong IP address was set on the proxy server "w", thus causing incorrect access control.
c) A wrong IP address was set on the router "w", thus causing incorrect access control.
d) The setting on the router "w" was wrong, causing communication using the HTTP protocol to be disabled.

Answer 1

Communication Network

Correct Answer

[Subquestion 1] e, g

[Subquestion 2] b

[Subquestion 3] b

Comments

This is a question concerning a communication network (LAN) connected to the Internet. There are intranets, company LAN systems using Internet technologies, and extranets, which connect companies with business partners and clients. In either case, if the network is connected to the Internet, it is important how security-related issues are resolved.

In order to prevent unauthorized access and to ensure security, a firewall (defense system) is constructed. The basic function of a firewall is to control data flow. In general, a router is set up or a proxy server is placed for that purpose.

[Subquestion 1]

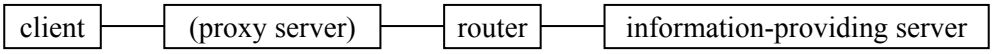
A router has the functions of the path-selecting (routing) and the filtering. The routing is the process of selecting the paths along which data is sent from a source node to a destination node in a network. The address information (an address table) of the nodes connected to the LAN is held on a router, and this information is used when routing. Filtering is the function that controls the data flow on a network. The subquestion asks about the "filtering of router h", so you can only look at the internal access relation between the information-providing server and the head office LAN clients. Each of the clients is managed by a proxy server, and they "always pass through proxy servers", so you can regard the source of communication from Client h1 as proxy server h. The destination, then, is the information-providing server. Communication from the information-providing server is just the opposite. Hence, the correct answers are (e) and (g).

The firewall in the answer group is only for communication access to the Internet. What communicates with "router h" is "proxy server h", not "client h1". Therefore, all other options are incorrect.

[Subquestion 2]

The company's internal information-providing service occurs between the clients and the information-providing server through the proxy servers. Focusing on the paths of this communication, we see that the four equipments involved are the clients, proxy servers, routers, and information-providing server. The positions of the proxy servers are already known (labeled), and they are connected to the clients and the routers. Since routers are connected to proxy servers and the information-providing server, (2) in Figure 2 cannot be a client; if it could, there would be a contradiction. A router is the unit that has functions only up to the network layer, so (2) is a router.

Therefore, we have



and the correct answer is (b).

[Subquestion 3]

According to the question text, the following information is provided:

(1) All clients were able to communicate with the information-providing server and the servers on the Internet.

(2) The new client that was added to the factory LAN was able to communicate directly with the other clients on the factory LAN.

The above indicates that the problem is not with the LAN or the client that was added. Communication from a client to the information-providing server or other servers on the Internet always goes through a proxy server, so the only common server for internal and external access is the proxy server. The client does not communicate directly to routers or servers beyond the proxy server; instead, the proxy server makes such communications on behalf of the client.

Hence, the proxy server is to control the access using the IP address set up in the proxy server. If the access is made denied by that setup, the proxy server rejects the requests, so the access control by the IP address set up in the proxy server w is erroneous. The correct answer, therefore, is (b). Communication between the newly added client and the existing clients is made within the same LAN, so they communicate each other directly without a proxy server. This is why communication on the same LAN is possible even though the proxy server w has incorrect settings. As for the other options in the answer group, here are some remarks:

a) If access control to the information-providing server is erroneous, it is possible to access the outside.

c) In communicating through router w, all client communication within the factory LAN passes through proxy server w, so adding a client does not make it necessary to set up an IP address at router w.

d) If communication following the HTTP protocol is disabled, no communication can take place from other clients to servers on the Internet or to the information-providing server.

5 Information Processing Technology

[Scope of Questions]
System performance, System reliability,
Risk management, Security, Standardization,
Operations research, etc.

Question 1

Q1. Read the following description about transaction processing performance, and then answer Subquestion.

As shown in the figure below, there is a server which consists of one CPU and two disks. As shown, databases DB1 and DB2 are allocated to different disks. In the case of this server, different transactions for each disk can be executed completely independently. Table 1 gives the breakdown for the average time of the use of system resources per transaction based on the result of the transaction-processing analysis.

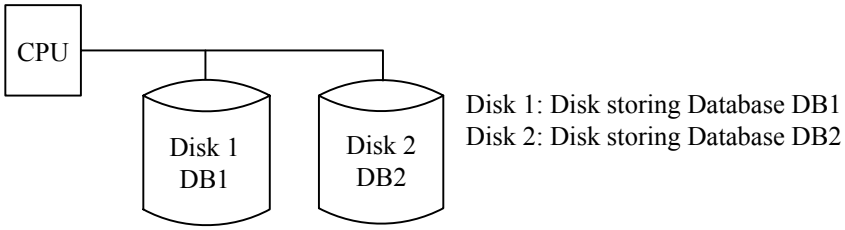


Fig. Server Design

Table 1 Average Time of Use of System Resources per Transaction

System resource	Average time of use per transaction
CPU	20 ms
Disk 1	80 ms
Disk 2	40 ms

The number of transactions processed by the server per second at any given point in time is called the TPS (Transactions Per Second) of that server. While operating, the TPS of the server, the usage rate of a given resource, and the average time of use of system resources per transaction (abbreviated hereinafter merely as "average time of use") are related as follows:

Usage Rate = TPS × Average Time of Use …….. [1]

For example, the TPS of the server is given by the following formula when the average time of use of a given system resource is 50 ms and the usage rate for that resource is 80%.

$$\text{TPS} = \frac{0.8}{0.05} = 16$$

Subquestion

From the answer groups below, select the correct answers to be inserted in the blanks [ ] in the following descriptions.

(1) When the upper limit on the usage rate of each system resource shown in the figure is placed at 40%, the upper limit for the TPS of this system is [A].

(2) When we continue to assume that the upper limit on the usage rate of each system resource is 40%, in order to quadruple the upper limit on the TPS of the system, it is necessary to reduce the access time for the disks by at least [B]. Here, it is assumed that when the system resource is made n times as large as before, the system resource access time in the transaction processing is reduced to 1/n.

(3) When the TPS of this server is 10, the average response time to process a transaction is approximately [C] times as long as the transaction processing time. Here, it is assumed that the average response time for each system resource is as follows, and that the average response time for transaction processing is the sum of the average response time for each system resource.

$$\text{Average Response Time} = \frac{\text{Average Time of Use}}{1 - \text{Usage Rate}} \quad \ldots\ldots\ldots [2]$$

(4) Based on the above formulas [1] and [2], one can see that, as the TPS of the server goes up, the usage rate of system resources increases and [D].

Answer group for A:

a)	0.005	b)	0.01	c)	0.02
d)	5	e)	10	f)	20

Answer group for B:

a) increasing the number of Disk 1 to four.
b) increasing the number of Disk 2 to four.
c) increasing the number of Disk 1 to two, and the number of Disk 2 to two.
d) increasing the number of Disk 1 to two, and the number of Disk 2 to four.
e) increasing the number of Disk 1 to four, and the number of Disk 2 to two.

Answer group for C:

a) 0.48 b) 0.95 c) 1.66
d) 2.86 e) 3.51

Answer group for D:

a) the average response time increases
b) the average response time decreases
c) the average time of use increases
d) the average time of use decreases

Answer 1

Transaction Processing Performance

Correct Answer

A – d, B – e, C – e, D – a

Comments

The theme of this question is performance calculation of transaction processing, but, in general, such calculations as usage rate and average response time are classified under queuing theory. However, calculation formulas and procedures are all described in the question text, so prior knowledge on queuing theory is not necessary to answer this question. The point is whether you can carry out the calculations accurately based on the explanations in the question text.

(1) Concerning TPS, the number of transactions per seconds, the question states that TPS = 0.8 / 0.05 = 16 given that the average time of use is 50 ms (= 0.05 sec.) and the usage rate is 80%. From this, it is clear that TPS = usage rate / average time of use. Now, it is given that the usage rate is 40%, and the average time of use for each device is given in table 1, so we calculate the TPS for each device using these values and obtain the following.

System Resource source	TPS
CPU	0.4 / 0.02=20
Disk 1	0.4 / 0.08=5
Disk 2	0.4 / 0.04=10

If a process that can be carried out in parallel consists of multiple processing units, the processing performance of the entire system is adversely affected by the processing unit with the worst performance. Such a processing unit is called a bottleneck. In this case, Disk 1 is the bottleneck, and the upper limit for the TPS as a server is answer (d) 5.

(2) To quadruple the upper limit for the TPS of the server means to change the current value of 5 to 20. The TPS of the CPU is 20, so this has passed, but the TPS of Disks 1 and 2 are 5 and 10, respectively, both well below the target; therefore, some performance improvement is required. The strategy for this performance improvement, indicated in the question text, is to increase the number of the system resource (disk). The performance of each disk unit increases in proportion to the number of the disk, since the assumption is that, if the number of the resource becomes n times, the access time decreases to 1/n.

Some phrases in the question text might be a little confusing, but we assume that the access time is equal to the average time of use. (Access time = usage rate / TPS.)

First, to increase the TPS of Disk 1 to 20, while keeping the upper limit at 40% of its usage rate, the access time needs to be made 0.02 sec. (= 0.4 / 20). This access time is 1/4 of

the current 0.08 sec., so the number of Disk 1 must be quadrupled to 4. Similarly, for Disk 2, the current access time of 0.04 sec. needs to be made 0.02 sec., so the number of the disk must be doubled to 2. Hence, the correct answer is (e), increasing the number of Disk 1 to four and the number of Disk 2 to two.

(3) Transaction processing time is the total time (sum of the durations) necessary for the processes that make up the transaction and does not include the waiting time. On the other hand, the average response time includes the sum of the transaction processing time and the waiting time. The contents of the transaction process in this question are not known, but the question text states that the "average response time for transaction processing is the sum of the average response time for each system resource". Hence, we may similarly assume that the transaction processing time is the sum of the average time of use for each system resource. Thus, the transaction processing time, according to Table 1, is 20 + 80 + 40 = 140 ms. The average response time needs to be calculated for each system resource, using Equations [1] and [2].

CPU: Usage Rate = 10 × 0.02=0.2, Average Response Time = 0.02 / (1 – 0.2) = 0.025 = 25 ms.
Disk 1: Usage Rate = 10 × 0.08=0.8, Average Response Time = 0.08 / (1 – 0.8) = 0.4 = 400 ms.
Disk 2: Usage Rate = 10 × 0.04=0.4, Average Response Time = 0.04 / (1 – 0.4) = 0.067 = 67 ms.

Hence, the average response time for the server is 25 + 400 + 67 = 492 ms. The question is how many times this value is larger than the transaction processing time (140 ms), so the answer is 492 / 140 = 3.514, which gives the correct answer (e).

(4) From Equation [1], which states that Usage Rate = TPS × Average Time of Use, we see that, when TPS increases, the usage rate also increases. Equation [2] is as follows:

$$\text{Average Response Time} = \frac{\text{Average Time of Use}}{1 - \text{Usage Rate}}$$

From this formula we see that, when the usage rate increases the average response time increases as well. Hence, the answer is (a). If these formulas are not enough to convince you, consider a specific situation, where, say, the usage rate goes up from 0.2 to 0.4. The average time of use could be anything, but say it is 1. If the usage rate is 0.2, then the average response time is 1 / (1 – 0.2) = 1.25. If the usage rate then changes to 0.4, then the average response time becomes 1 / (1 – 0.4) = 1.67.

[Addendum concerning Queue]

To answer this question, prior knowledge of queue is not necessary. In fact, the question hints on how to answer questions involving queue. Since this is an exercise, it is a good idea to learn queuing theory from this question. Here, additional explanation is provided for your reference.

It is a common experience to wait at supermarkets and banks. A similar concept applies to computer processing. We evaluate computer performance from two aspects: processing time from the standpoint of the computer and response time from the standpoint of the user. The concept of queuing theory is applied to evaluate the latter, the response time.

Recall that response time is the duration between the time when a request is made and the time when the response to that request is obtained. In the real world, for example, at a teller window in a bank, it falls into the time from the moment when you start standing in a waiting line until you finish receiving service. This (response) time, when you begin waiting in the line, consists of the time it takes all of the people already in the line to be served (i.e., the time you spend waiting) and the time you are being served.

Response time = amount of time you spend waiting
+ amount of time you are receiving service

The first of these, the amount of time you spend waiting, is called waiting time, and the response time is the sum of this waiting time plus the amount of time you are receiving service. The waiting time is determined by the length of the line and the time it takes each person to be served. Here, the time it takes each person to be served can be determined by skill, etc. In fact, a question on queuing theory basically pre-estimates the length of the line.

Now, from your experience, consider checkout lines at the supermarket. When do they become long? It happens when the supermarket becomes crowded. But is that it? Even if two lines consist of the same number of people and are formed at the same time, when one cashier is inexperienced and takes more time, the line will be longer. In other words, the length of a waiting line is determined by the number of customers and the performance (skill) of the cashier. The concept that can represent both of these at the same time is usage rate. Usage rate is a representation of how much of the ability is being used, so it is a value that can be obtained by skill and the number of customers.

Based on this idea, let us obtain the usage rate. If the time of use of a CPU is 20 ms, then it can process 50 transactions per second. If there are 10 transactions, the usage rate is 10 / 50 = 0.2. In this exercise question, the number of transactions per second (TPS) is multiplied by the time of use in order to obtain the usage rate. The value TPS × (time of use) shows for how long that resource is used in one second, so its meaning is indeed the usage rate. However, one must be careful with units. The unit of time for transactions is 1 second, so the time of use should also be in seconds. If everything were converted to milliseconds, we would have TPS × (time of use) = 10 × 20=200, which does not make sense since the usage "rate" should be no more than 1.

In generally used models, it is known that the length of the queue is obtained by (usage rate) / (1 – usage rate). So we use this equation to find the length of the queue. The processing time of this length of the queue (the number of processes in the queue) is the waiting time, so we calculate the waiting time by (length of the queue) × processing time (time of use). In the example above, the time of use was 20 milliseconds, and the usage rate was 0.2. Let us assign

these numbers into the equation. The length of the queue is 0.2 / (1 – 0.2) = 1/4. You may be bothered that it is not an integer value, but recognize that one person waits once every four times, etc. So the waiting time is 1/4 of the processing time, i.e., 1/4 × 20 milliseconds = 5 milliseconds. To this waiting time, we add the 20 milliseconds required for one process. The total response time then is 25 milliseconds. Incidentally, Equation [2] is also listed in the question text

$$\text{Average Response Time} = \frac{\text{Average Time of Use}}{1 - \text{Usage Rate}}$$

If you assign the average time of use (20 milliseconds) and a usage rate of 0.2 into this equation you also get 20 / (1 – 0.2) = 25 milliseconds, the same answer, naturally. The response time is the sum of the length of the queue and the processing time of one transaction. We get the length of the queue as follows:

$$\text{Length of the Queue} = \frac{\text{Usage Rate}}{1 - \text{Usage Rate}}$$

We get the number of processes in the response time as follows:

$$\text{The Number of Processes in the Response Time} = \frac{\text{Usage Rate}}{(1 - \text{Usage Rate})} + 1$$

$$= \frac{\text{Usage Rate}}{1 - \text{Usage Rate}} + \frac{(1 - \text{Usage Rate})}{1 - \text{Usage Rate}}$$

$$= \frac{\text{Usage Rate} + (1 - \text{Usage Rate})}{1 - \text{Usage Rate}} = \frac{1}{1 - \text{Usage Rate}}$$

And to get the response time, we multiply this by the processing time (time of use)

$$\frac{\text{Time of Use}}{1 - \text{Usage Rate}}$$

Now, you may be wondering why there is even a waiting time when the usage rate is less than 1. For instance, with Disk 1 in this question, the usage rate is 0.8 when the TPS is 10. If we calculate the length of the queue, we get 0.8 / (1 – 0.8) = 4. The disk access time is 80 milliseconds with a TPS of 10, i.e., one transaction occurs every 100 milliseconds. So, the disk should have 20 milliseconds after each access, and you may think waiting would not occur. In a sense, this is a correct line of thinking. Here, the assumption is that access times and transactions occur at fixed intervals. If this is the case, no waiting occurs. However, the model of this queue is based on the assumption that access times as well as transactions occur randomly without certain patterns. But if they are completely random, we could not do any calculations, so we assume that we do know at least the average. Note that all values given in the question have the term "average". This is a key word. The time of use is the average time of use, and the response time is the average response time. In this explanation, the term "average" is omitted, while they are all supposed to be "averages", to be precise. For your information, in all queue models that appear on exam questions, the number of transactions per unit of time is assumed to have a Poisson distribution and the processing time per transaction has a distribution called an exponential distribution.

Question 2

Q2. Read the following description about scheduling, and then answer Subquestion.

A subcontractor is selected in order to complete a certain project in the minimum number of days at the lowest possible cost. This project can be divided into eight tasks, *A* through *H*, and each of these tasks is given a specific order of precedence, as shown in Table 1. If a certain task has a precedent task, it cannot be started until the precedent tasks are completed.

Table 2 shows the number of working days and costs if these tasks are entrusted to subcontractor *X* or *Y*. Subcontractor *X* can complete each task at less cost than subcontractor *Y*, though the former requires more working days than the latter.

Table 1 List of tasks

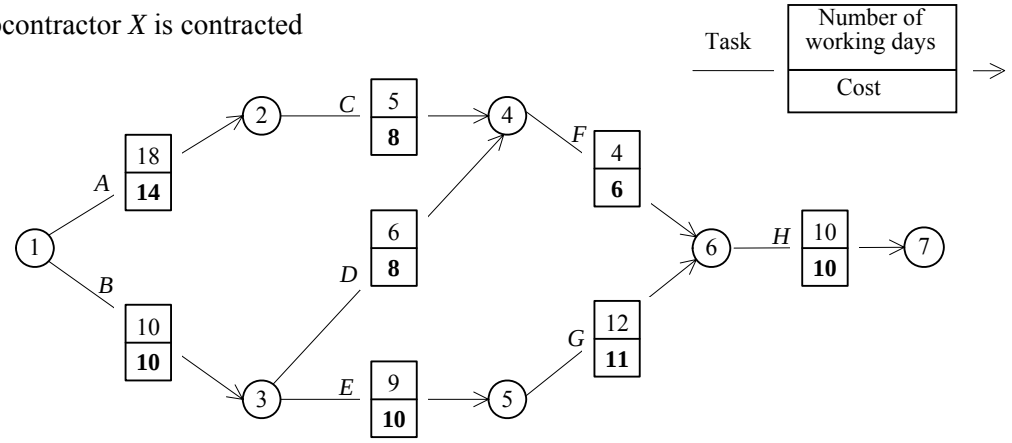
Task	Precedent task
A	-
B	-
C	*A*
D	*B*
E	*B*
F	*C*, *D*
G	*E*
H	*F*, *G*

Table 2 Number of working days and costs by subcontractor

Task	Subcontractor *X*		Subcontractor *Y*	
	Number of working days	Cost	Number of working days	Cost
A	18	14	16	15
B	10	10	8	13
C	5	8	4	10
D	6	8	5	10
E	9	10	8	13
F	4	6	3	9
G	12	11	10	12
H	10	10	9	12

Each of the arrow diagrams on the next page shows the case in which subcontractor *X* or *Y* is contracted to perform all the tasks.

If subcontractor *X* is contracted



If subcontractor *Y* is contracted

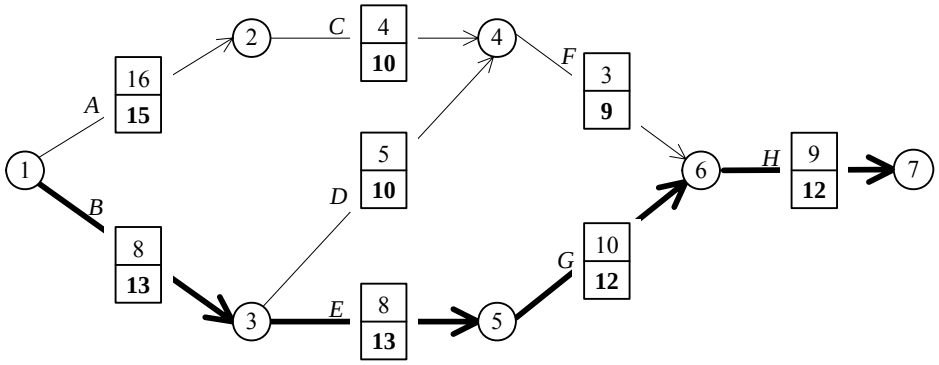


Figure Arrow diagram of the project

Subquestion

From the answer group below, select the correct answers to be inserted in the blanks [ ] in the following description.

(1) When subcontractor *X* is contracted to perform all the tasks, the cost needed to complete the project is [A] and the number of days needed is [B].

(2) When subcontractor *Y* is contracted to perform all the tasks, the cost needed to complete the project is 94 and the number of days needed is 35. The critical path for this schedule is *BEGH*, shown by the thick line in the diagram for subcontractor *Y*. If subcontractor *X* is contracted to perform part of the tasks, the number of working days remains the same (35 days) and the cost can be decreased; specifically, the lowest cost will be [C].

Answer group

a) 35	b) 41	c) 63	d) 74	e) 77
f) 86	g) 87	h) 88	i) 89	j) 90

Answer 2

Schedule Planning

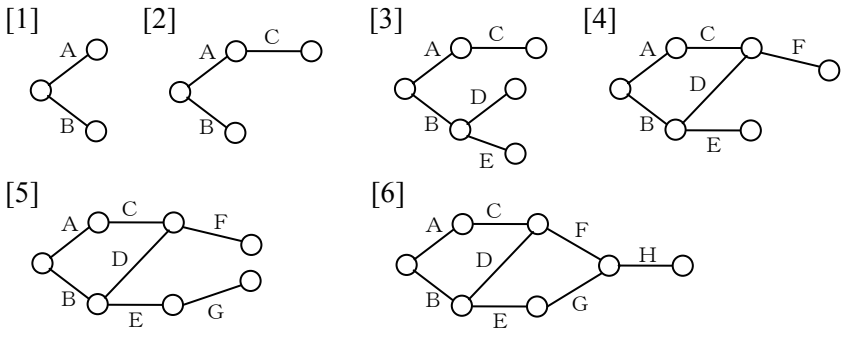
Correct Answer

A – e, B – b, C – g

Comments

This is a question on selecting a subcontractor so that a project can be completed in a minimum number of days at the lowest possible cost.
This particular project can be divided into eight tasks labeled *A* through *H*, and they have precedence requirements, as shown in Table 1 of the question text. Below is a figure indicating these relations, which develop into a PERT diagram.

[1] *A* and *B* have no precedence requirements, so they can be done in parallel.
[2] *C* has *A* as a precedent (*C* must be preceded by *A*), so *C* must come after *A*.
[3] *D* and *E* must be preceded by *B*, so they come after *B*.
[4] *F* must be preceded by *C* and *D*, so *F* must come after *C* and *D*.
[5] *G* must be preceded by *E*, so *G* must come after *E*.
[6] *H* must be preceded by *F* and *G*, so *H* must come after *F* and *G*.



A: The cost of hiring subcontractor *X* to perform all the tasks is simply the total sum of the costs of *X* from Table 2. It is (e): 14 + 10 + 8 + 8 + 10 + 6 + 11 + 10 = 77.

B: The number of days required for completion should be obtained using the PERT diagram. The number of tasks is rather small, so it does not take much time even if you have to consider all of the possible paths.

- $A \rightarrow C \rightarrow E \rightarrow H$ (37 days)
- $B \rightarrow D \rightarrow F \rightarrow H$ (30 days)
- $B \rightarrow E \rightarrow G \rightarrow H$ (41 days)

Hence, the number of days required for completion is 31 + 10 = 41 days (b). The critical path is $B \rightarrow E \rightarrow G \rightarrow H$.

C: The necessary cost of hiring subcontractor *Y* to perform all the tasks is the total sum of the costs of *Y* from Table 2, which is 15 + 13 + 10 + 10 + 13 + 9 + 12 + 12 = 94, and the number of days required for completion is 35. The question now is to keep this 35-day schedule but reduce the cost by hiring subcontractor *X* to perform part of the tasks.

From the text of the question and Table 2, we know that for each task, subcontractor *X* takes more days than subcontractor *Y* but the cost of *X* to perform the task is less than the cost of *Y* to perform the same task. From this condition, we see that the cost can be reduced if we hire subcontractor *X* to perform the task(s) that has some margin in the schedule and does not affect tasks that follow.

Since the tasks of the critical path are $B \rightarrow E \rightarrow G \rightarrow H$, which take 35 days altogether, we need to choose from the tasks on the other paths, namely *A*, *C*, *D*, and *F*, to be assigned to subcontractor *X*. Here, all of the tasks up to node [6] must be completed within 35 – (days necessary for *H*) = 35 – 9 = 26 (days). Further, because the number of additional days and costs both depend on the tasks, we put priorities on the tasks that are highly cost-effective per day. Consider, as shown below, the difference between subcontractors *X* and *Y* for each of the tasks *A*, *C*, *D*, and *F*:

Task	# of days	Cost	Cost per day
Task *A*	+2	-1	-0.5
Task *C*	+1	-2	-2
Task *D*	+1	-2	-2
Task *F*	+1	-3	-3

So first let us consider hiring *X* to do task *F*. The number of days increases by 1, but the cost gets reduced by 3.

If F is assigned to *X*, the tasks up to node [4] must be finished within 26 – (number of days for *X* to do *F*) = 26 – 4 = 22 (days). Here, if we hire subcontractor *X* to perform tasks *C* and *D* also, the number of days increases only by 1 for each of the tasks, so we can still complete these within 22 days and, at the same time, reduce the cost by 2 for each of the tasks.

Finally we consider task *A*. Now that task *C* has been assigned to subcontractor *X*, all the tasks up to node [2] must be completed within 22 – (number of days for *X* to do *C*) = 22 – 5 = 17 (days). However, if task *A* is to be assigned to subcontractor *X*, the number of days up to that point becomes 18, so we cannot do this.

Therefore, the minimum cost, when subcontractor *X* is hired to perform tasks *C*, *D*, and *F*, will be

(the cost when *Y* is hired to do all the tasks) – (cost reduction)

= 94 – (3 + 2 + 2)

= 94 – 7 = 87.

The answer then is 87, which is (g).

Question 3

Q3. Read the following description about the floating point number representation, and then answer Subquestions 1 through 3.

Represent a number in the 32-bit floating point representation as shown in Figure 1.
The meaning of each bit is as follows:

(1) Bit 0: Sign
This bit contains the sign of the mantissa. "0" indicates a positive value, and "1" indicates a negative value.

(2) Bits 1 through 7: Exponent
These bits contain the binary expression of the exponent in base two. Negative values are expressed in two's complement.

(3) Bits 8 through 31: Mantissa
These bits contain the binary expression of the absolute value of the mantissa. Bit 8 contains the first bit in the fractional part.

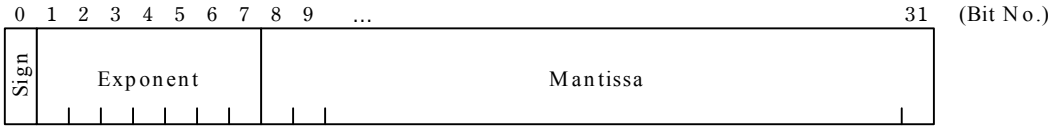


Fig. 1 Format for Floating Point Representation

Subquestion 1

Figure 2 illustrates a method for converting a binary value in floating point representation to a decimal value. Select the decimal values to be inserted in the blanks [A] and [B] from the answer groups below.
Here, "$(0.11101)_2$" in Figure 2 indicates that "0.11101" is a binary number.

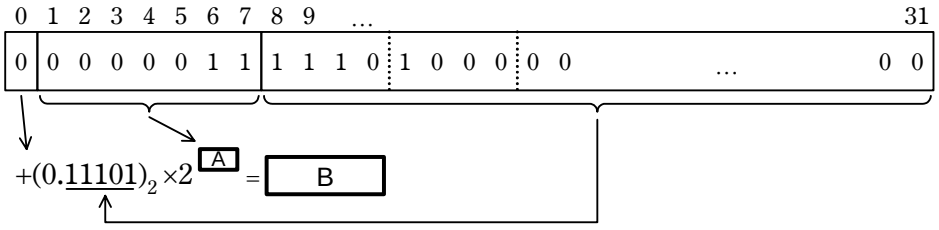


Fig. 2 Method for Converting a Binary Number in Floating Point Representation to a Decimal Value

Answer group for A:

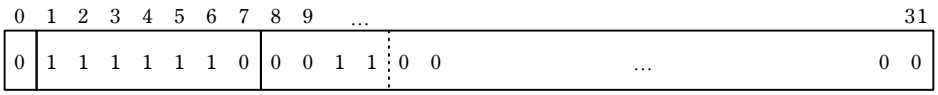
a) 0 b) 1 c) 2 d) 3 e) 4

Answer group for B:

a) 0.725 b) 0.74 c) 7.25
d) 7.4 e) 72.5 f) 74

Subquestion 2

From the answer group below, select the correct decimal representation of the following binary number in floating point representation.



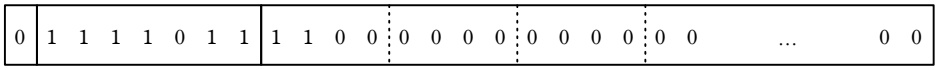
Answer group:

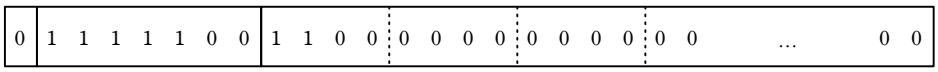
a) 3×2^{-6} b) 3×2^{-4} c) 3×2^{-1} d) 3×2^{0}
e) 3×2^{1} f) 3×2^{2} g) 3×2^{122}

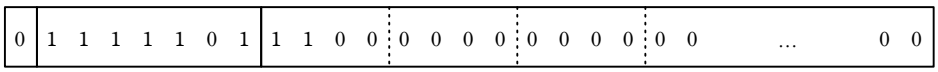
Subquestion 3

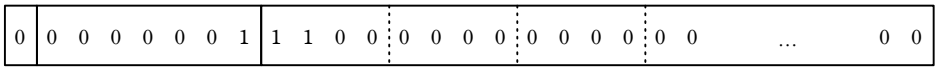
Select the normalized representation of the floating point number in Subquestion 2 from the answer group below. Here, "normalizing" means manipulating the exponent and the mantissa to set the leftmost bit of the mantissa (Bit 8) to "1".

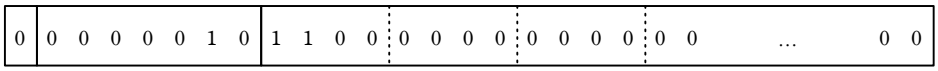
Answer group:

a) 

b) 

c) 

d) 

e) 

Answer 3

Floating Point Representation

Correct Answer

[Subquestion 1] A – d, B – c

[Subquestion 2] a

[Subquestion 3] b

Comments

This is a question on floating-point representation. Almost every Morning Examination has a question on floating-point representation. Yet, a vast majority of examinees seem to be poor at this. Most questions on floating point given in the Common FE Examination can be answered if you just understand radix conversion and fixed-point number, so it is imperative that you learn these concepts.

Floating point is a way to represent numerical values by their order (an approximate size of the number, like "thousands" or "millions") and the leading digits of a certain length (mantissa). In fixed-point representation, the necessary number of digits to be represented (hence the number of bits) depends on the order of the numerical value, but computers handle all the numbers in the constant length internally, so the range of numerical values that can be represented is limited. On the other hand, when a wide range of numbers must be handled, not all digits are equally important; often, the leading digits of a certain length are more important. It can therefore be said that even if some errors may be contained in terms of accuracy, it is more convenient to be able to handle a wider range of numerical values. Floating point is a method of numerical expression that meets these needs and demands, able to express a wide range of numerical values (with very large and very small absolute values) with a certain length of digits (number of bits).

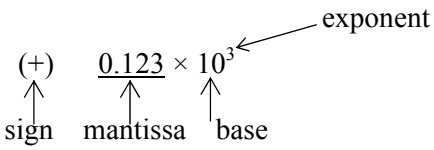
Let us take decimal numbers (+) 123 for our example of floating point number.

As we move the position of the decimal point, we get:

$$
\begin{aligned}
&\vdots \\
&= 1230 \times 10^{-1} \\
(123)_{10} \quad &= 123 \times 10^{0} \\
&= 12.3 \times 10^{1} \\
&\vdots \\
&= \boxed{0.123 \times 10^{3}} \quad \cdots\cdots [1] \\
&= 0.0123 \times 10^{4} \\
&\vdots
\end{aligned}
$$

As shown above, by changing the position of the decimal point, one numerical value can be expressed in infinite ways. To set up a unified standard, we make it a rule to use the expression labeled [1] above. This is called normalization. Normalization is a manipulation to have more effective digits by putting the leftmost nonzero number in the first decimal place. By deciding in advance what base (radix) to use (in this case, 10), we can determine each numerical value by its sign, mantissa, and exponent (the power of the radix number for the order).

In the above example, we have the following:



Inside the computer, these are processed as the sign (0 for a positive number, 1 for a negative number), the exponent, and the mantissa, all expressed in the binary system. For the exponent, there are two formats:

(1) fixed-point format (two's complement)
(2) biased expression (excess expression)

The first format is not used very often, but questions involving this format can appear on the exam. In contrast, the second format is an expression used in real computers. This format is thoroughly explained in the question text.
The conversion is done as follows:

(1) The decimal number is converted to the binary number by radix (base) conversion.
(2) Normalize the binary number so that the first nonzero number (1 in binary) comes to the first decimal place.
(3) Express the exponent value in two's complement.
(4) The sign is put to the sign section, the exponent to the exponent section, and the mantissa to the mantissa section.

For the decimal number used in the example above, note that

$(123)_{10} = (1111011)_2$

and its floating-point expression is the following:

$(+)\ (1111011)_2 \times 2^0 = (+)(0.1111011)_2 \times 2^7$

[Subquestion 1]

Because the exponent is in two's complement, $(0000011)_2$ represents "+3". Therefore, the value expressed by Figure 2 is

$(+)\ (0.11101)_2 \times 2^3$
$= (111.01)_2 \times 2^0$
$= 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$
$= (7.25)_{10}$

The correct answer for blank A is (d), and the correct answer for blank B is (c).

[Subquestion 2]

From the bit expression $(1111110)_2$ of the exponent section, since the leading bit is 1, we see that the value is negative. This is in two's complement with 7 bits (fixed point format), so the value is "-2". Therefore, the value shown for this subquestion is "$(+)\ (0.0011)_2 \times 2^{-2}$", which can be expressed as follows.

$(+)\ (0.0011)_2 \times 2^{-2}$
$= (+)\ ((11)_2 \times 2^{-4}) \times 2^{-2}$
$= (+)\ (11)_2 \times 2^{-6}$
$= (+)\ (3)_{10} \times 2^{-6}$

The answer, therefore, is (a).

[Subquestion 3]

As explained above, normalization is such manipulation of a number that the leading nonzero number comes to the first decimal place. Hence, the number in Subquestion 2 can be expressed as follows.

$(+)\ (0.0011)_2 \times 2^{-2}$
$= (+)\ ((0.11)_2 \times 2^{-2}) \times 2^{-2}$
$= (+)\ (0.11)_2 \times 2^{-4}$

Hence, the correct answer should be the one whose exponent section expresses "-4" in two's complement. "+4" is $(0000100)_2$, so using two's complement, we get $(1111100)_2$. The answer is (b). For your information, the exponents expressed by other options are (a) -5, (c) -3, (d) +1, and (e) +2.

6 Algorithms

[Scope of Questions]
Sort, Search, Character string processing, File processing, Diagram, Graph, Numeric calculation, etc.

Question 1

Q1. Read the following program description and the program itself, and then answer Subquestion.

[Program Description]

Function `RadixConv` is a program that converts a digit string in base `M` (2 <= `M` <= 16) to a digit string in base `N` (2 <= `N` <= 16).

(1) A base `M` digit string consists of base `M` digits alone without space characters. In the case of base 11~16, alphabetic characters "A" ~ "F" are used to denote decimal values 10 ~ 15.

(2) `RadixConv` first converts a base `M` digit string to an integer and then converts it to a base `N` digit string. Function `MToInt` converts a base `M` digit string to an integer and function `IntToN` converts the integer to a base `N` digit string.

(3) Function `MToInt` and function `IntToN` use the following functions:

(i) Function `ToInt` that converts a numerical character `P` ("0", "1", …, or "F") to an integer

(ii) Function `ToStr` that converts an integer `Q` (0 <= `Q` <= 15) to a numerical character ("0", "1", …, or "F")

(iii) Predefined function `Length` that returns the length of the string

(iv) Predefined function `Substr` that extracts part of the string

(4) Tables 1 through 5 below list the specification of arguments and return values of the functions.

Table 1 `RadixConv`

Argument/ Return value	Data type	Meaning
`Frdx`	Integer	Radix of digit string before conversion (2<= `Frdx` <= 16)
`Fnum`	Char	Digit string before conversion
`Trdx`	Integer	Radix of converted digit stringe (2 <= `Trdx` <= 16)
Return value	Char	Converted digit string in base `Trdx`

Table 2 `MToInt`

Argument/ Return value	Data type	Meaning
`Rdx`	Integer	Radix of digit string before conversion (2 <= `Rdx` <= 16)
`Num`	Char	Digit string before conversion
Return value	Integer	Converted integer

Table 3 `IntToN`

Argument/ Return value	Data type	Meaning
`Val`	Integer	Integer before conversion
`Rdx`	Integer	Radix of converted digit string (2 <= `Rdx` <= 16)
Return value	Char	Converted digit string in base `Rdx`

Table 4 `ToInt`

Argument/ Return value	Data type	Meaning
`P`	Char	A numerical character before conversion ("0", "1", …, or "F")
Return value	Integer	Converted integer

Table 5 `ToStr`

Argument/ Return value	Data type	Meaning
`Q`	Integer	Integer before conversion (0 <= `Q` <= 15)
Return value	Char	Converted numerical character

[Program]

```
○ char_type: RadixConv (int_type: Frdx, char_type: Fnum, int_type: Trdx)
• return IntToN(MToInt(Frdx, Fnum), Trdx)
                                    /* Takes IntToN value as return value of function */

○ int_type: MToInt(int_type: Rdx, char_type: Num)
○ int_type: Idx, Val
• Val ← 0
■ Idx: 1, Idx <= Length(Num), 1  /* Length returns string length of Num */
  • Val ← [ A ] + ToInt(Substr(Num, Idx, 1))
■   /* Substr returns the Idx^th(>= 1) character from the beginning of Num */
• return Val                     /* Takes Val as the return value of function */

○ char_type: IntToN(int_type: Val, int_type: Rdx)
○ int_type: Quo                  /* Quotient */
○ int_type: Rem                  /* Remainder */
○ char_type: Tmp
• [ B ]
• Tmp ← ""
■ Quo >= Rdx
  • Rem ← Quo % Rdx
  • Tmp ← ToStr(Rem) + Tmp/* + is an operator to concatenate strings */
  • [ C ]
■
• [ D ]
• return Tmp                     /* Takes Tmp as return value of function */

○ int_type: ToInt(char_type: P)
○ int_type: Idx
○ char_type: Code[16]            /* Index begins at 0 */
/* Code stores initial values "0", "1", "2", "3", "4", "5", "6", "7",      */
/* "8", "9", "A", "B", "C", "D", "E", "F"  in this order                   */
/* Character values are incremented in this order                          */
• Idx ← 0
■ [ E ]                          /* Compare strings */
  • Idx ← Idx + 1
■
• return Idx                     /* Takes Idx as the return value of function */

○ char_type: ToStr(int_type: Q)
○ char_type: Code[16]            /* Index begins at 0 */
/* Code stores initial values "0", "1", "2", "3", "4", "5", "6", "7",      */
/* "8", "9", "A", "B", "C", "D", "E", "F"  in this order                   */
/* Character values are incremented in this order                          */
• return Code[Q]                 /* Takes Code[Q] as the return value of the function */
```

Subquestion

From the answer groups below, select the correct answers to be inserted in the blanks ☐ in the above program.

Answer group for A:

a) `Rdx`
b) `Val`
c) `Val * Rdx`
d) `Val / Rdx`

Answer group for B, C, and D:

a) `Quo ← Quo / Rdx`
b) `Quo ← Quo / Rem`
c) `Quo ← Rdx`
d) `Quo ← Rem / Rdx`
e) `Quo ← Val`
f) `Rem ← Rdx`
g) `Rem ← Val`
h) `Tmp ← ToStr(Quo) + Tmp`
i) `Tmp ← ToStr(Rem) + Tmp`

Answer group for E:

a) `P < Code[Idx]`
b) `P > Code[Idx]`
c) `P <= Code[Idx]`
d) `P >= Code[Idx]`

Answer 1

Radix Conversion Program (Pseudo Language)

Correct Answer

A – c, B – e, C – a, D – h, E – b

Comments

In [Program], five small programs (functions) are shown in sequence. This question format is different from that of the previous questions, so it may have been startling. However, the important thing is to look into a program by dividing it into several parts. Let us approach this question in this way.

Every program, no matter how big it is, is made up of a combination of smaller functions necessary to accomplish the whole function. The more complicated functions the program has, the more these smaller functions it needs, resulting in a large program. However, if you only focus on the smaller functions, you can consider them without being overwhelmed by the entire program size. In general, when thinking about a complicated question, it is recommended to begin by removing the cause of the complexity. When the size is the cause, you should divide the whole into smaller parts. When the function is complicated, you may think that the smaller functions necessary to accomplish the function are intricately intertwined. However, as we have learned in the sections related to module cohesion and coupling, program design is considered favorable when the modules (smaller functions) are highly independent of each other. On examination questions, there are hardly any programs against this design policy, so you need not worry so much.

First, the function (purpose) of the entire program needs to be understood. At the beginning of [Program Description], its function is clearly described as follows: "Function `RadixConv` is a program that converts a base `M` digit string to a base `N` digit string". We must be sure to keep the function in mind as we answer this question, not failing to "see the forest for the trees." But at the same time, be careful not to focus too much on the "forest" (the entire program).

Next, we try to understand the overview of each of the sections (functions). Every question always comes with clues; you have to find them. If a program is clearly divided into subroutines or functions, the clues are in the arguments and return values. Keeping the function of the entire program in mind, let us look into an overview of each function.

- **`RadixConv`** (Table 1)

  This function itself is the purpose of the entire program. Though shown already, let us review how it is organized and see what is involved. The arguments are the radix of digit string before conversion (`Frdx`), a digit string before conversion (`Fnum`), and the radix of converted digit string (`Trdx`). The return value is the converted digit string. Hence, we see that this section is to convert a numeral string (`Fnum`) in base `M` (value of `Frdx`) to a new base `N` (`Trdx`) string and to return the numeral string which is the result of the conversion. This is the purpose of the program as a whole.

- **`MToInt`** (Table 2)
  The arguments are the radix of digit string before conversion (`Rdx`) and a digit string before conversion (`Num`); the return value is the converted integer. Hence, this is a function that converts the pre-conversion numeral string (`Num`) to an integer (numerical value). As the data types of the arguments indicate, the numeral string is of the character type. In general numeral strings cannot be used for calculations, so any calculation requires conversion of numeral strings into the corresponding numerical values. Also the name of this function "`MToInt`" suggests "conversion from `M` to an integer", so one can guess what this function does from its name.

- **`IntToN`** (Table 3)
  The arguments are an integer before conversion (`Val`) and the radix of converted digit string (`Rdx`), and the return value is the converted digit string. Based on this, this is a function that converts the pre-conversion integer (`Val`) to a numeral string in the base `N` (`Rdx`) expression. The name "`IntToN`" probably means "conversion from an integer to `N`".

- **`ToInt`** (Table 4)
  The argument is a numerical character (`P`) before conversion, and it returns the converted integer. The numeral (`P`) is converted to an integer. Note that even a numerical character is considered a character (string).

- **`ToStr`** (Table 5)
  The argument is an integer (`Q`) before conversion, and the return value is a converted numerical character. This converts an integer (`Q`) to a numerical character. You may be bothered by the fact that there is no radix in the argument. However, regardless of what radix (base) is used, the value corresponding to the numerical character is the same. ,What radix is used determines the range of numerical values that the argument `Q` may take. For example, in base 2, this must be 0 or 1; in base 16, it is in the range 0 to 15. In either case, the same numerical value corresponds to the same numeral.

  While examining these functions, you may have wondered what radix is used for integers (numerical values) before and after conversion. However, numerical values are values which are independent of the radix (the base used). When we say "base-n system", we are just referring to the numeration system in which we express numerical values (if you must think of the radix in this case, think 2 because everything inside the computer is expressed in binary). When converting a numeral string in base-7 into base-9, we indeed convert it to base-10 temporarily; however, we are not really converting it to base-10 but to a numerical value. The easiest numeration system for us to understand is base-10, so we convert the number to base-10 to understand its value.

Blank A:

This function "`MToInt`" is the one that converts a numeral string in base `M` to an integer. From the last line "`return Val`" we see that the integer value after conversion is stored in `Val`. The function itself is simply a repetition of the process whose condition is "`Idx <= Length(Num)`". The initial value of `Idx` is 1; "`Length` (`Num`)" indicates the length of the numerical string `Num`, and its increment is 1, so it is repeated as many times as its number of the characters. "`Substr(Num, Idx, 1)`" found in the repetition means to "take out the one character in the

`Idx`th position (from the beginning) of `Num`". So, the program is taking one character at a time from the beginning of the numeral string `Num` and adding the value to `Val` which will be the result of the conversion.

This conversion of a base-`M` number to an integer value is equivalent to conversion to base 10. Recall how this is being carried out. For example, consider the binary number 101. This is $1\times2^2(=4) + 0\times2^1(=0) + 1\times2^0(=1)$, so this calculation gives the value 5 (in base 10). In other words, we take the numeral of each digit, multiply the weight corresponding to the bit position, and find the sum. But the answer group does not seem to include this method. Note that $1\times2^2 + 0\times2^1 + 1\times2^0 = ((1\times2)+0)\times2+1$, so this latter method is what is being used here. "2" corresponds to `M` in base `M`, so we can multiply `Val` by `Rdx` (which contains this value `M`) and add the numeral values of each bit position. Therefore, the correct answer is (c). Recall that the function `ToInt` is to convert a one-character numeral to an integer (value).

If you notice that $1\times2^2 + 0\times2^1 + 1\times2^0 = ((1\times2) + 0)\times2+1$, the answer is not so difficult to find. But even if you did not, you need not give up. Use the fact that the correct answer is in the answer group. When the arguments `Rdx` = 2 and `Num` = "101" are given to this function, it is already known that `Val` is 5. When assigning each of the answer options into the blank and tracing the program, the one that gives the correct result is the answer. In this example, `Num` has 3 digits, so the process is done 3 times as follows ([①] ~ [③] indicates the number of repetitions).

a) : ① `Val` = 2 + 1 = 3 → ② `Val` = 2 + 0 = 2 →③ `Val` = 2 + 1 = **3**

b) : ① `Val` = 0 + 1 = 1→② `Val` = 1 + 0 = 1 →③ `Val` = 1 + 1 = **2**

c) : ① `Val` = 0 × 2 + 1 = 1 →② `Val` = 1 × 2 + 0 = 2 →③ `Val` = 2 × 2 + 1 = **5**

d) : ① `Val` = 0 / 2 + 1 = 1 →② `Val` = 1 / 2 + 0 = 0.5 →③ `Val` = 0.5 / 2 + 1 = **1.25**

When you know what the correct result should be, as shown in this case, trying each answer option to find the result is an effective method. As for the manipulation $1\times2^2 + 0\times2^1 + 1\times2^0 = ((1\times2) + 0)\times2+1$, this is an effective way conceived to reduce the amount of calculation, and this has appeared in both Morning and Afternoon Examinations. Take this opportunity to learn it.

Blanks B - D:

Recall that function `IntToN` is one that converts an integer to a numeral string in base `N`. It is a bit longer compared to the function `MToInt`, and there are more blanks, but do not let that scare you. Be confident. First, note that, the return value is the converted numeral string, and its content is found in `Tmp`. As far as the structure is concerned, the repetition process is the main part, and we see that the program is concatenating the characters (numerals) in `Tmp` while remainders are obtained in each step. With this in mind, remember how radix conversion is done. For example, to convert the numerical value 5 to binary system, recall that the number is divided by 2 repeatedly while keeping track of the remainder each time.

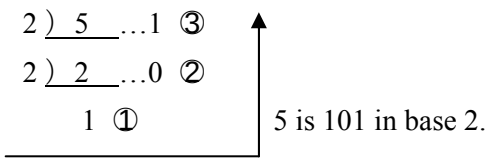


Figure 1

Do not just stare at the algorithm written in pseudo language. It is important that you keep examples such as the one above in mind.

For blank B, then, you can figure out that this is probably variable initialization, considering its position relative to the following sentence (`Tmp ← ""`). In variable initialization, the point is to identify a variable that needs to be initialized for the sections that follow. The condition for repetition is "`Quo >= Rdx`", and `Rdx` is an argument, so its value is already set. But "`Quo`" (quotient) is a variable defined in this function, so initialization is necessary before comparison. Thus, we now understand that this is initialization for the variable `Quo`.

Next, consider what `Quo` is. At the beginning of the repetition, we find "`Rem ← Quo % Rdx`"; the remainder obtained when `Quo` is divided by `Rdx` is stored in the variable `Rem`. ("`%`" is the modulo operator, but you can figure this out even if you do not know this, because `Rem` is the remainder.) Considering this and the example given in Figure 1, you should be able to see that this must be the part where the sequence of changes 5 → 2 is made. If this is the case, the comment "quotient" given to this variable also makes sense. The initial value of it is the integer before conversion, so the correct answer is (e) `Quo ← Val`. Note that the argument `Val`, which is an integer before conversion, is not used anywhere. This argument is a necessary value but is found nowhere in the function. This means that the value is transferred to another variable. This too is a clue that guides you to the correct answer.

For blank C, consider the contents of the repetition simply. First, "`Rem ← Quo % Rdx`" places the remainder into `Rem`, resulting from division by `Rdx` which is the "`N`" in base-`N`. The next sentence "`Tmp ← ToStr(Rem) + Tmp`" concatenates the remainder to the variable `Tmp` after converting to a numeral. Here, `Tmp` is a variable where a resulting numeral string is stored. In the next repetition, the value of `Quo` needs to be changed, so the part corresponding to this change must be inserted here. In the example of Figure 1, this is where "5 → 2" takes place. In the next repetition, the value of `Quo` must be the quotient obtained when the number is divided by `Rdx`, but this quotient is missing. Even if you search for where the quotient is already obtained, this cannot be found, either. Hence, it is necessary to find the quotient here and substitute the value into `Quo`. Hence, the correct answer is (a) `Quo ← Quo / Rdx`.

In blank D, the rest of the repetition must take place. Using the example above, `Tmp` now contains the numeral string corresponding to the procedures ② ③ of Figure 1, (i.e., "01") so far. So, the result of the last step ① must be concatenated, and this one is actually a quotient, not a remainder. Also, the condition for stopping the division repetition is that the quotient (1) is smaller than the divisor (2). This matches the repetition condition for this function as well. Therefore, we see that at this point it is necessary to convert the quotient to a numeral and to concatenate it to the front of the character string. The correct answer is (h) `Tmp ← ToStr(Quo) + Tmp`. There is also a clue for this blank in terms of the structure of the program. The only thing that happens after the blank is that the return value is returned. Hence, no matter what variable value is changed other than the variable `Tmp`, the return value is not affected. This suggests that the line has to involve the variable `Tmp`, narrowing the choice for the correct answer to either (h) or (i). Then, you should identify the correct answer.

Blank E:

Recall that function `ToInt` is to convert its argument, a one-character numeral, to an integer. From the answer options and the name of variable `Idx`, you have probably figured out that some table search is involved. Since the return value is `Idx`, you should know that this index (`Idx`) is

returned whenever the table is searched sequentially and the condition is satisfied. The answer group is divided by the condition for continuing the repetition: while the argument `P` is smaller (a, c) or while it is greater (b, d). First, let us decide which of these is correct. The table "`Code`" being searched has characters "0" through "F" in ascending order of the character code. The initial value of this search is `Idx` = 0, so the search begins at the first element. In other words, the search is difficult to hold a condition where `P` is smaller. Now that we know that the correct inequality is either > or >=, we have only to decide if = is needed or not; this will give us the correct answer. For instance, when the argument is "0", we must have `Idx` = 0. However, when the inequality is (d) "`P` >= `Code[Idx]`", the first comparison already satisfies the condition, so `Idx` is added, resulting in `Idx` = 1. Therefore, the correct answer should not have =, and that is (b) `P` > `Code[Idx]`.

At the beginning of this question, we looked briefly at all the functions; however, in an actual examination, you should probably keep the blanks in mind and follow the program when it becomes necessary to do so.

You may have noticed that this question contained many instances that remind you of characteristics of a program in C, such as using return values of functions without substituting them into variables. Terms such as "`ToInt`" and "`ToStr`" are definitely like C. This trend is likely to continue, so everyone should have at least some basic knowledge of the C language. However, you do not have to know the syntax in detail, nor do you have to be able to write programs.

Question 2

Q2. Read the following program description, pseudo-language syntax explanation, and the program itself, and then answer Subquestions 1 and 2.

[Program Description]

This is a Sub-program `QuickSort` which sorts the integers from `A[Min]` to `A[Max]`(0 <= `Min` < `Max`) in one-dimensional array `A`.

(1) Sorting procedures are as follows.

(i) The program searches the array from `A[Min+1]` to `A[Max]` sequentially for an element whose value is different from the value of `A[Min]`, take the first such element found, compare it with A[`Min`], and select whichever is the larger as the reference value (`Pivot`). If all of the elements in the array are the same, sort processing ends. For selecting the reference value, a sub-program `FindPivot` is used.

(ii) Elements are rearranged so that all elements less than the `pivot` are `A[Min]`, …, `A[i-1]` (`Min` < `i` <= `Max`) and all elements equal to or greater than the `pivot` are `A[i]`, …, `A[Max]`. A sub-program `Arrange` performs this processing.

(iii) The rearranged elements (`A[Min]`, …, `A[i-1]` and `A[i]`, …, `A[Max]`) are treated as two new arrays and sorted by recursively applying `QuickSort`.

(2) Argument specifications for the sub-programs are given in the following tables.

Table 1 `QuickSort` Arguments

Variable name	Input/Output	Meaning
`A`	Input/Output	One-dimensional array to be sorted
`Min`	Input	Index of first element in sort range
`Max`	Input	Index of last element in sort range

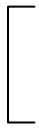
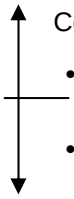
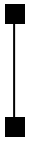
Table 2 `FindPivot` Arguments

Variable name	Input/Output	Meaning
`A`	Input	One-dimensional array to be sorted
`Min`	Input	Index of first element in sort range
`Max`	Input	Index of last element in sort range
`Ret`	Output	Returns index of element storing pivot value. However, returns `"-1"` if `A [Min]`, …, `A [Max]` are all the same value.

Table 3 `Arrange` Argument

Variable name	Input/Output	Meaning
`A`	Input/Output	One-dimensional array to be sorted
`Min`	Input	Index of first element in sort range
`Max`	Input	Index of last element in sort range
`Pivot`	Input	Reference value
`Ret`	Output	Rearranges elements so values `A[Min]`, …, `A[i-1]` are less than `Pivot` and values `A[i]`, … `A[Max]` are equal to or greater than `Pivot`, and returns the value of `i`.

[Pseudo-Language Syntax Explanation]

Syntax	Explanation
	A continuous area where declarations and processes are described.
○	Declares name, type and other attributes of procedures, variables, etc.
• Variable ← Expression	Substitutes expression value for variable.
{statement}	Writes comment.
	Indicates selection. When the conditional expression is true, process 1 is executed. When false, process 2 is executed.
	Indicates repetition with termination condition at the top. While the conditional expression is true, the process is executed.

[Program]

```
○ Sub-program name: QuickSort(A[], Min, Max)
○ Integer: Pivot, J, K, L

• FindPivot(A[], Min, Max, J)
J > -1
  • Pivot ← A[J]
  • Arrange(A[], Min, Max, Pivot, K)
  • L ← K - 1
  • QuickSort(A[], Min, [ A ])
  • QuickSort(A[], [ B ], Max)
```

```
○ Sub-program name: FindPivot(A[], Min, Max, Ret)
○ Integer: Pivot, K
○ Logical: Found

• Pivot ← A[Min]
• K ← Min + 1
• Ret ← -1
• Found ← false
■ K <= Max  and  not Found
    A[K] = Pivot
      • K ← K + 1
    ---------------
      • Found ← true
      A[K] > Pivot
        • [ C ]
      ---------------
        • [ D ]
■
```

```
○ Sub-program name: Arrange(A[], Min, Max, Pivot, Ret)
○ Integer: L, R, Tmp

• L ← Min
• R ← Max
■ L <= R
  • Tmp ← A[L]
  • A[L] ← A[R]
  • A[R] ← Tmp
  ■ A[L] < Pivot
    • L ← L + 1
  ■
  ■ A[R] >= Pivot
    • R ← R - 1
  ■
■
• Ret ← L
```

Subquestion 1

From the answer groups below, select the correct answers to be inserted in the blanks ☐ in the above program.

Answer group for A and B:

a) `K` b) `L` c) `Max` d) `Min`

Answer group for C and D:

a) `Ret ← A[K]` b) `Ret ← A[Max]` c) `Ret ← A[Min]`
d) `Ret ← K` e) `Ret ← Max` f) `Ret ← Min`

Subquestion 2

The contents of arguments have been summarized in Table 4 after using `QuickSort` to sort elements `A[0]` to `A[9]` of an integer-type one-dimensional array. From the answer groups below, select the correct answers to be inserted in the blanks ☐ in the following table.

Table 4 `QuickSort` Call Count and Content of arguments

Call count cycle	A	Min	Max
1st cycle	A[0] A[9]: 3 5 8 4 0 6 9 1 2 7	0	9
2nd cycle	3 2 1 4 0 6 9 8 5 7	0	4
3rd cycle	E	F	G
⋮	⋮	⋮	⋮
nth cycle	0 1 2 3 4 5 6 7 8 9	9	9

Answer group for E:

a) 0 1 2 4 3 6 9 8 5 7

b) 0 2 1 3 4 6 9 8 5 7

c) 0 2 1 4 3 6 9 8 5 7

d) 0 4 1 2 3 6 9 8 5 7

Answer group for F and G:

a) 0 b) 1 c) 2

d) 3 e) 4 f) 5

Answer 2

Quick Sort (Pseudo Language)

Correct Answer

[Subquestion 1] A – b, B – a, C – d, D – f

[Subquestion 2] E – c, F – a, G – c

Comments

Quick sort appears in the Morning Examination, where it may be defined in such a way as this: "It is the method of sorting data in which the entire set of data is first divided into two groups, one with elements smaller than a certain value and the other with elements greater than or equal to the value. The same grouping process is applied again to each of the two groups, and this process is repeated until all the data are sorted". This method is efficient as the number of comparisons and switchings is small, but many people may have the feeling that the recursive nature of the process makes it difficult to handle. Many examinees, when they saw this question on the examination, might have thought to themselves, "Oh no! I knew I should have studied quick sort algorithms". **Remember, though, that these examinees are trapped and taken in by the question the moment they feel this way.** Examination questions are designed with the intent that any examinee with general knowledge can answer them, so it is crucial that you maintain composure as you answer these questions.

[Subquestion 1]

This is a fill-in-the-blank question, but whenever the specific procedures are described in [Program Description] like this question, it is best to first get an overview of the program and then consider its correspondence with [Program Description]. Just like the question theme "Quick Sort", whenever a question involves an algorithm that everyone couldn't come up with easily, the question tends to appear in this way.
According to [Program Description], the sorting procedures by the subprogram `QuickSort` are as follows:

(i) Use the subprogram `FindPivot` to select a reference value (`Pivot`).
(ii) Use the subprogram `Arrange` to rearrange the elements in the array (`A[Min],…, A[i - 1]` are less than `Pivot, A[i],…, A[Max]` are greater than or equal to `Pivot`).
(iii) Consider `A[Min],…, A[i - 1] and A[i],…, A[Max]` as new arrays, and apply `QuickSort` recursively.

First, let us pay attention to the names of the subprograms used in this section and consider their correspondence with the subprogram `QuickSort`; the following is not hard to identify.

```
○ QuickSort(A[], Min, Max)
○ Integer: Pivot, J, K, L
・ FindPivot(A[], Min, Max, J)                    …… (i)
▲ J > –1
  ・ Pivot ← A[J]
  ・ Arrange(A[], Min, Max, Pivot, K)             …… (ii)
  ・ L ← K - 1
  ・ QuickSort(A[], Min, [     A     ])
  ・ QuickSort(A[], [     B     ], Max)           …… (iii)
▼
```

Blanks A and B:

We proceed with explanations assuming that we know the recursive calls of `QuickSort` containing these two blanks, corresponding to (iii). Table 1 lists the arguments of `QuickSort`; the three arguments are "`A` (the array to be sorted), `Min` (the index of the first element in the range), and `Max` (the index of the last element in the range)" in that order. Consider what to insert into these blanks, given the contents of these arguments. Blank A must be the index of the last element of the range, corresponding to "`i - 1`" in (iii) of the description. Blank B is the index of the first element in the range, corresponding to "`i`" in (iii). Now, concerning these two numbers, "`i - 1`" and "`i`", we read in (ii) of the description, "all elements less than `Pivot` are `A[Min]`, …, `A[i - 1]` and all elements equal to or greater than `Pivot` are `A[i]`, …, `A[Max]`". Hence, "`i - 1`" is the index of the last element less than `Pivot` while "`i`" is the index of the first element greater than or equal to `Pivot`. The elements of the array to be sorted, `A[Min]`, …, `A[Max]`, are not listed in this order at the beginning. By `FindPivot` of (i), a value `Pivot` is chosen, and by `Arrange` in (ii), the elements are sorted. Take a look at the arguments of `Arrange` in Table 3. There are five arguments, but the last item `Ret` says "rearranges elements so values `A[Min]`, …, `A[i - 1]` are less than `Pivot` and values `A[i]`, …, `A[Max]` are equal to or greater than `Pivot`." In the program, `K` is used instead of this `Ret` (note "`Arrange(A[],Min,Max,Pivot,K)`"), so this value `K` corresponds to `[i]` referred to in (ii). Hence, Blank A should be "`K - 1`", but the answer group does not contain this choice. Looking back at the program, we see that there is a line `L ← K – 1` immediately after `Arrange` is called. So `L = K – 1`, and the correct answer for blank A is (b) `L`. It should be evident by now that the correct answer for blank B should be (a) `K`.

Blanks C and D:

These blanks are located in the subprogram `FindPivot`, which is explained in [Program Description] (i). The description (i) is a little long, but its contents can be summarized as follows: "Search through the array from `A[Min+1]` to `A[Max]` sequentially for an element whose value is different from `A[Min]` (this includes all values except `A[Min]`), take the first such element found, compare it with `A[Min]`, and select whichever is the larger as the reference value (selection of `Pivot`) ". Now, take a look at the answer group. Note that all of the choices set some value to `Ret`. Concerning this variable `Ret`, Table 2 explains it as an argument for `FindPivot`: it "returns index of element storing reference value". We then see that these blanks are to determine the reference value and set its index in the argument `Ret`. The section containing these blanks is executed when the condition "`A[K] = Pivot`" is false. Meanwhile, at the beginning of the program, we read "`Pivot ← A[Min]`", so at this point "`Pivot = A[Min]`". Therefore, that the condition "`A[K] = Pivot`" is false means that the value of `A[K]`

is different from the value of A[Min]. True or false of the condition "A[K] > Pivot " determines the value of the argument Ret, this section is exactly the part that corresponds to the "selection of Pivot" described above. Blank C applies when A[K] > Pivot (= A[Min]). Recall that Pivot is the larger value of A[K] (different from A[Min]) and A[Min], and here A[K] is larger, so A[K] becomes Pivot. Blank C is then (d) Ret ← K. In contrast, blank D is the other case, so the correct answer is (f) Ret ← Min. Answers (a) through (c) all set one of the elements of array A as the argument Ret. By carefully reading the explanation in Table 2 ("returns index of element storing pivot value"), the fact that these are wrong answers should be immediately clear. Hence, you have three choices left. Quick sort sounds like a difficult concept, but hopefully you have been convinced that it is actually not such a difficult question.

[Subquestion 2]

This is a tracing question. The blanks in the program are already filled in, so basically there is no choice but to insert the contents of the blanks and trace the program carefully. This is the subquestion that tests you on the heart of this question, which is quick sort, i.e., recursion.

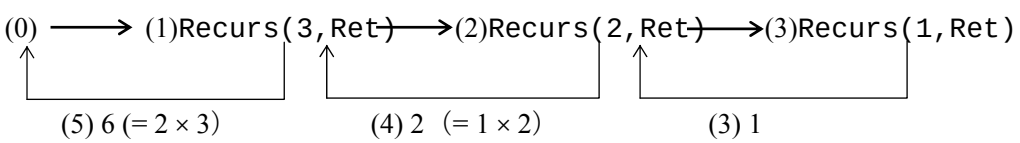
Let us first review recursion briefly. Recursion means to "call oneself". An example often used to explain recursion is the factorial function. The program is written as follows. Recall that the factorial (of a number) is the product of integers from 1 to that number. For example, factorial 3 (3!) is 1 × 2 × 3 = 6.

[Example]

```
○ Recurs(n, Ret)
○ integer: n, Ret
  n > 1
   ・ Recurs(n-1, Ret)
   ・ Ret ← n × Ret
  ──────────────
   ・ Ret ← 1
```

(In pseudo language used on examination questions, it seems that functions themselves cannot have return values, so it looks different from a program actually written in a programming language like C.)

Now, consider how "Recurs(3,Ret)" is obtained using this program. First, since the argument is 3, the value of the argument n is “3”, which satisfies "3 > 1", so "Recurs(2,Ret)" is called. The argument n is 2, and because "2 > 1", it calls "Recurs(1,Ret)". Next, since the argument n is 1, and "1 > 1" is false, the argument Ret is set to 1, and the program returns to where the call was made. At the source of the call (Recurs(2,Ret)), the value of this Ret (= 1) is used to execute "Ret ← Ret (= 1) × n (= 2)". Then it returns to the source of the call (Recurs(3,Ret)) again, with "2" as the value of Ret. This enables the program to complete the process of finding "Recurs(3,Ret)" as it executes "Ret ← Ret (= 2) × n (= 3)". Consequently, Ret becomes 6, the correct result. The following diagram shows how the result is obtained.



As seen here, the program calls itself, but in order for the process to end, that which was called last must end.

Now, let us consider the program given in the question. As Table 4 suggests, `QuickSort(A[],0,9)` is called first. Next, the second call occurs, calling `QuickSort(A[],0,4)`. At this point it appears that the first call divided array `A` into `A[0]`, …, `A[4]`, which are less than the `Pivot` value, and `A[5]`, …, `A[9]`, which are greater than or equal to the `Pivot` value. Hence, what follows this grouping (sorting by `Arrange`) is, on the surface, calling of "`QuickSort(A[],0,4)`" and "`QuickSort(A[],5,9)`". However, note that the calls do not continue in this order necessarily. As mentioned earlier, if during "`QuickSort(A[],0,4)`", which is executed first, another recursive call is made, that is executed first.

`QuickSort(A[],0,9)` ← First call

…

`FindPivot` and `Arrange` divide the array into two groups: `A[0]`,…, `A[4]` and `A[5]`,…, `A[9]`.

…

`QuickSort(A[],0,4)` ← Second call

`QuickSort(A[],5,9)` ← May not be the third call

Calls made during `QuickSort(A[],0,4)` are executed first.

Let us trace the second call (`QuickSort(A[],0,4)`). The range subject to sorting here is `A[0]`, …, `A[4]`, and the value of each element is shown below (from Table 4).

0	1	2	3	4
3	2	1	4	0

First, consider `Pivot`, which is obtained by `FindPivot`. The first value different from `A[0]` (=`A[Min]`) =3, i.e., the larger of `A[1]=2` and `A[0]=3`, so `A[0] = 3,` is chosen. Next, setting `Pivot` = 3, we now trace `Arrange(A[],0,4,3,K)`.

The beginning of the program has "`L ← Min` " and "`R ← Max`", so `L` = 0, and `R` = 4. Then, the program goes into repetition while the condition `L <= R` is satisfied.

(i) There are three instructions first: "`Tmp ← A[L]`", "`A[L] ← A[R]`", and "`A[R] ← Tmp`". Are all of these processes right? This is a standard method of switching two values. In other words, values of `A[0]` and `A[4]` are switched, resulting in `A[0] = 0` and `A[4] = 3`.

(ii) In the next repetition, while `A[L] < Pivot`, the value of `L` is increased by 1 each time. Since `A[0] = 0, A[1] = 2, A[2] = 1, A[3] = 4`, we see that `A[3] < Pivot` does not hold, resulting in `L = 3`.

(iii) Next, in the repetition below that, while "`A[R] >= Pivot`", the value of `R` is decreased by 1 each time. Since `A[4] = 3, A[3] = 4, and A[2] = 1,` we see that `A[2] >= Pivot` does not hold, resulting in `R` = 2.

(iv) Because `L = 3` and `R = 2`, the repetition condition is not satisfied, completing the subprogram. The diagram below shows the processes thus far.

Pivot		L	R	0	1	2	3	4
				3	2	1	4	0
3	(i)	0	4	0				3
	(ii)	0→3		0	2	1	4	
	(iii)		4→2			1	4	3
	(iv)	3	2	0	2	1	4	3

When the repetition ends, the value of L (= 3) at that point is set to Ret, and the program returns to where the call was first made.

Now let us consider the blanks. First, look at blank E. A[0] through A[4] are in condition (iv) in the above chart. A[5] through A[9] have not changed, so we inherit the same condition from the second call (in all of the choices in the answer group, this part is identical). Hence, the correct answer is (c). Next, consider blanks F and G. Tracing makes it clear that the value of Ret (= K) is 3 and thus A[0], …, A[2] are the values less than Pivot while A[3] and A[4] are the values greater than or equal to Pivot. With these results, the second call (QuickSort(A[], 0, 4)) is made, during which two further calls "QuickSort(A[], 0, 2)" and "QuickSort(A[], 3, 4)" are made. Hence, the third call is "QuickSort(A[], 0, 2)", so the answer for blank F is (a) 0, and the answer for blank G is (c) 2.

You should now be able to figure out how the answers are obtained. How does that feel? If you studied quick sort algorithms prior to the test, you probably feel more comfortable and confident, and perhaps you have some advantage. However, even if you did not, this question is not at all an impossible one to answer as long as you know this topic at the level that appears on the Morning Examination (the process of recursively repeating the grouping procedure in which data are divided up into two groups, those greater than a certain value and those less than or equal to the same value) and understand the calling nature of a recursive process. Algorithm questions on the Common FE Examination have always emphasized the ability to trace rather than the ability to construct an algorithm; but in recent years, this ability to trace programs has become all the more important. In this explanation we traced the second call, but it is a good idea to trace the first call as well as the third and following calls. For your reference, we conclude this section by presenting a tracing chart on the first call below.

Pivot	L	R	0	1	2	3	4	5	6	7	8	9	Remarks
			3	5	8	4	0	6	9	1	2	7	
5	0	9	7									3	switching
	0		7										addition of L
		9										3	addition of R
	0	9	3									7	switching
	0→1		3	5									addition of L
		9→8									2	7	addition of R
	1	8		2							5		switching
	1→2			2	8								addition of L
		8→7								1	5		addition of R
	2	7			1					8			switching
	2→5				1	4	0	6					addition of L
		7→4					0	6	9	8			addition of R
	5	4	3	2	1	4	0	6	9	8	5	7	L > R

7 Program Design

[Scope of Questions]
System development process,
Program design process, Structured design,
Module design,
Program design document, etc.

Question 1

Q1. Read the following description about program design, and then answer the Subquestions 1 through 4.

A system is designed that provides domestic news based on users' preferences. The outline of this system is as follows.

[System Outline]

(1) News to be provided is saved in a news file in the following format. The news file is a sequential file consisting of records each of which contains a news item, and the news items are recorded in the order they were registered.

Record Format of News File

Date of registration	Time of registration	Date of occurrence	Time of occurrence	Category	Head-line	Out-line	Details	Address of image file

The date of registration is an eight-digit character string representing the year, month and day when the news item was registered. For example, "20080401" represents April 1, 2008. The time of registration is a four-digit character string representing hours and minutes when the news item was registered. For example, "1830" represents 6:30 p.m. The date of occurrence is the year, month and day when the news event occurred and has the same format as that of the date of registration. The time of occurrence is the time when the news event occurred and has the same format as that of the time of registration. The category is a category of news, and stores one of the following: "Healthcare," "Education," "Economy," "Entertainment," "Science," "Society," "Sports," and "Politics." In addition, the headline for the news item, its outline, its details, and the address of a related image file are included.

(2) User information is registered in the users file in the following format. The users file is an indexed file using the user ID as a key.

Record Format of Users File

User ID	Category 1	Category 2	Category 3	Category 4	Category 5	Date of last use	Time of last use

The users file registers up to 5 categories of each user's preferred news in category fields 1 through 5. If less than five categories, "NIL" (indicates empty) is stored in the remaining category field(s). The system provides at a maximum of 10 news items backwards by date and time for each registered category.

The date of last use and time of last use are the date and time when the user used the system last. The date of last use and the time of last use are character strings in the same

formats as the date of registration and the time of the registration respectively.

(3) If more than 24 hours have elapsed since the date and time of the last use, all of the news items registered within the past 24 hours are subject to extraction. If the previous use of the system by a user is within 24 hours, only news items which were registered after the time of the last use are subject to extraction.

(4) Based on the type of the user's terminal, the system changes the format of extraction results to be provided. If the user's terminal is a cell-phone, the applicable terminal type is "Simple," and the system provides an outline of each news item. If the user's terminal is a PC, the applicable terminal type is "Detail," and the system provides details of each news item and associated images.

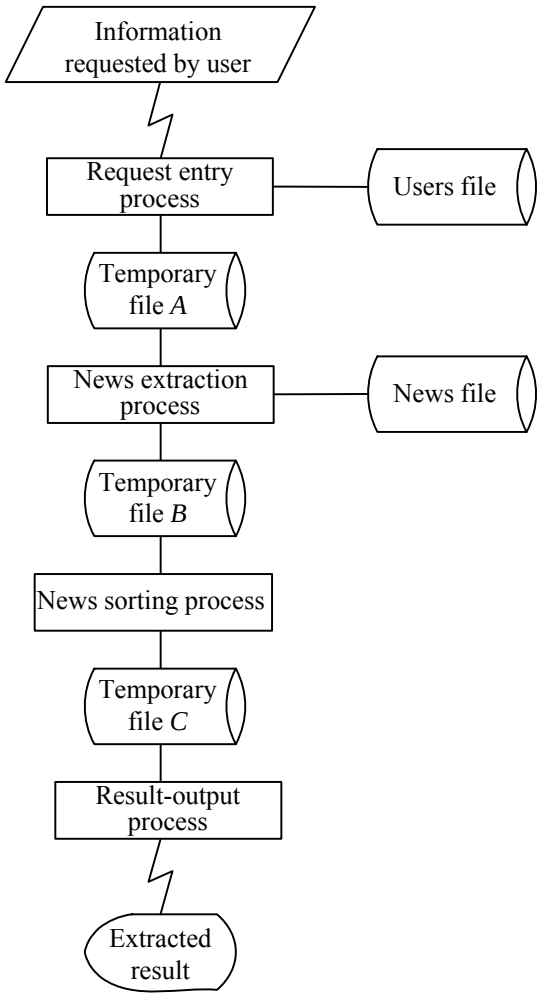


Fig. Process Flow

[Process Description]

(1) In the request entry process, a record whose user ID is the same as the one included in the requested information is selected from the users file. The requested information from the user has the following format.
The relevant category, "Simple" or "Detail," is stored in the terminal type field.

Format of Requested Information

User ID	Terminal type

Records in the following format are written into temporary file *A* from the requested information and the selected record.

Record Format of Temporary File *A*

Category 1	Category 2	Category 3	Category 4	Category 5	Date of extraction start	Time of extraction start	Terminal type

Finally, the date of last use and the time of last use in the relevant user record in the users file are updated to the current date and time.

(2) In the news extraction process, based on temporary file *A*, the record satisfying the following two conditions is simultaneously extracted from the news file.
   (i) The date and time of registration of the news record in the news file are more recent than the date and time of starting extraction from temporary file *A*.
   (ii) The category of news file records meets one of category fields 1 through 5 in temporary file *A*. Necessary information is added to the record satisfying these conditions, and then the record is written to temporary file *B*.

(3) In the news sorting process, temporary file *B* is sorted and then written to temporary file *C*.

(4) In the result-output process, up to 10 items are extracted for each category from temporary file *C*, and the results are written in the following format in accordance with the terminal type.

Format used when the terminal type is “Simple”:

Date of occurrence	Time of occurrence	Category	Headline	Outline

Format used when the terminal type is “Detail”:

Date of occurrence	Time of occurrence	Category	Headline	Detail	Address of image file

(5) Here, information between processes is transferred only using the files shown in the figure.

Subquestion 1

From the answer group below, select the correct answers to be inserted in the blanks [ ] in the following description.

A user's record in the users file is shown below:

XR205	Economy	Enter-tainment	NIL	NIL	NIL	20080401	2038

If this user used the system at 17:00 on April 3, 2008, [A] is stored in the Time-of-extraction- start field in the temporary file *A* by the request entry process, and [B] is stored in the Time-of-extraction-start field.

Answer group:

a) "0000"　　b) "1700"　　c) "2038"
d) "20080401"　　e) "20080402"　　f) "20080403"

Subquestion 2

The record format of temporary file *B* written by the news extraction process is shown below. From the answer group below, select the correct answers to be inserted in the blanks [ ] in this format.

Date of occurrence	Time of occurrence	[C]	Headline	Outline	Detail	Address of image file	[D]

Answer group:

a) Terminal type　　b) Time of extraction start
c) Date of extraction start　　d) Time of registration
e) Date of registration　　f) Category
g) User ID

Subquestion 3

From the answer group below, select the correct answers to be inserted in the blanks [ ] in the following description.

In the news-sorting process, records obtained in the news extraction process are subject to sort. In this case, the first sort key is [E], and the records are sorted in ascending order of the character codes. The second sort key is [F], and the records are sorted in descending order. The third sort key is [G], and the records are sorted in descending order.

Answer group:

a) Outline
b) Address of image file
c) Detail
d) Time of extraction start
e) Time of registration
f) Date of registration
g) Time of occurrence
h) Date of occurrence
i) Category

Subquestion 4

From the answer group below, select the correct answers to be inserted in the blanks [ ] in the following description.

A user's record in the users file is shown as follows:

DT512	Politics	Economy	Sports	NIL	NIL	20080401	0400

The number of registered news items for each category at 11:00 on April 1, 2008, is as shown below. If this user uses the system at the same time on the same day, the number of records written to temporary file C by the news sort process is [H], and the number of news items displayed by the result-output process is [I].

Table Number of Registered News Items

Category	Number of registered items per hour										
	00:00 to 00:59	01:00 to 01:59	02:00 to 02:59	03:00 to 03:59	04:00 to 04:59	05:00 to 05:59	06:00 to 06:59	07:00 to 07:59	08:00 to 08:59	09:00 to 09:59	10:00 to 10:59
Healthcare	0	0	0	0	0	0	2	0	1	0	0
Education	0	0	0	0	0	1	0	1	0	0	0
Economy	0	0	0	1	0	2	2	6	8	2	4
Entertainment	1	0	1	0	1	2	0	1	2	1	1
Science	1	0	1	0	1	1	3	2	3	2	5
Society	0	1	2	0	1	0	2	3	5	4	3
Sports	0	0	0	1	0	0	1	1	0	3	2
Politics	0	1	0	0	0	1	4	2	3	6	3

Answer group:

a) 27 b) 28 c) 50 d) 53
e) 60 f) 98 g) 108

Answer 1

Providing Domestic News Based on the User's Preferences

Correct Answer

[Subquestion 1] A – e, B – b

[Subquestion 2] C – f, D – a (order irrelevant in C and D)

[Subquestion 3] E – i, F – h, G – g

[Subquestion 4] H – c, I – a

Comments

The theme of this question is program design of a system that provides domestic news based on the users' preferences. The use of the Internet has become so popular in recent years that most examinees can probably imagine, upon reading the description of this question, a system in which the user can register categories he/she is interested in so that news articles are displayed, from the most recent ones, only in the categories that the user has registered. Keep this concept in mind as you read the question text.

The text consists of [System Outline], Process Flow (diagram), [Process Description], and Subquestions 1 through 4. Basically, you should read this in order from the beginning, but you must exercise a little creativity in allocating time for reading questions. On a normal examination, perhaps it is good to follow the policy "Read the question text carefully, organize setup and other conditions neatly, and then answer the questions"; however, on the Common FE Examination, particularly on the Afternoon Examination, the examinees often run out of time. You need to keep this in mind and come up with a way to read questions efficiently. Speaking frankly, the reason you read a question is to answer the subquestions. Even if you spend a long time reading the question description before the subquestions, when you answer the subquestions, you will end up reading the description again just to be sure. With that in mind, you may as well read the question with the assumption that you will eventually read it again.

Concerning [System Outline], it probably suffices to know the following four points:

(1) description of the news file
(2) description of the users file
(3) description concerning the time the user used the service last and the range of news items to be displayed
(4) description concerning the user's terminal type

Of course, if more detailed information can easily enter your mind, there is no need to try not to look at it. If there are any numbers or conditions that catch your attention, go ahead and underline them. But there is no need at this point to try to understand or memorize them.

Next, we read [Process Description]. Follow the process flow in the given chart, as you read along. Again it suffices to know that this description explains in (1) through (4) each of the processes shown in the chart and states in (5) the fact that information between processes is exchanged only using the files shown in the chart.

[Subquestion 1]

The given figure shows the contents of a users file, and the question asks for the values of certain contents (starting date of extraction and starting time of extraction) of temporary file A prepared by the request entry process when the user uses the system.

First, read the record of the user. Then read [System Outline] (2), and understand that the user ID is "XR205", the user has registered two categories ("economy" and "entertainment"), and the last time the user accessed this system was at 20:38 on April 1, 2008. At this point, you may notice the description "the system provides at a maximum of 10 news items backwards by date and time for each registered category".

What is being asked involves the contents of a temporary file created by the request entry process, so next you need to read (1) of [Process Description]. There, you will see that temporary file A consists of the requested information (user ID, terminal type) and the record selected (contents of the users file corresponding to the user ID), but the description does not state how the starting date or time of extraction will be used. So you should continue to read on. Once you get to (2), you will see that the starting date and time of extraction are used as record-extraction conditions for the news file. In other words, the program compares the date and time of registration of record in the news file with the starting date and time of extraction, and extracts only those records registered later than the start of extraction ((2)(i)).

Here, news is extracted from the news file to display only those news items that satisfy certain conditions. From the meaning of the item, we see that this involves a condition regarding time. Remember that there is a description somewhat related to this in (3) of [System Outline], so read that once again. According to that point, the situations are different depending on whether the time when the user is accessing the system elapsed more than 24 hours after the time when the user last accessed the system. If you check this point for this particular user, it has been more than 24 hours, so all news items registered during the last 24 hours are subject to extraction. Hence, the starting date and time of extraction in temporary file A are those values referring to the time 24 hours before the current time.

The current time is when the user is using this system, which is 17:00 on April 3, 2008. The time 24 hours prior to this is 17:00 on April 2, 2008. Hence, blank A is (e) "20080402", and blank B is (b) "1700".

[Subquestion 2]

This question is on the record format of temporary file B, which is the output of the news extraction process. Since this is the output of the news extraction process, you should reread (2) of [Process Description]. There, you will see that this file is based on temporary file A and contains records extracted from the news file as well as some additional necessary information, but you will not know the detailed format of the records (item composition).

In general, an input file is a file that summarizes all data necessary for that process. Temporary file B is created to output news on the user's terminal in the final result-output process. So, think from the standpoint of whether all items necessary to create this output are available. This is not unique to this particular question. In any question where you are to fill in a blank in a file format, pay attention to the output contents (file or printed report) using that file and look for missing items.

There are two things you must be careful about at this point. One is the case where there is

another separate step between the step in which the output contents referred to are prepared and the step being considered. In such a case, the process of that separate step between the two must also be considered. The other thing is whether or not the value of that item can be found in the step where the file subject to consideration is being created.

Between the news extraction process where temporary file B is created and the result-output process where the output contents are prepared, there is a news-sorting process. But since this step involves only sorting, the file items and values are not changed. Hence, you need not worry about the news-sorting process, so compare the output contents of (4) with the record format of temporary file B. Then it should be obvious that the record format is lacking "category". There is nothing else that gives any clues. So after a few worrisome moments, you remember the description of (5) in [Process Description]. Because information exchange between processes is limited to the files, you begin to think, "Isn't there any other information necessary to create the output contents?" The output contents have different formats depending on the type of terminal, so you need the terminal type. Now, we have two candidates for the answer: "category" and "terminal type". If you can find out whether these contents can be prepared in the news extraction process, then you can find the correct answer. Both of these items are contained in temporary file A, which is the input file, so both are fine. Therefore, the correct answers are (f) "category" for blank C and (a) "terminal type" for blank D. Of course, the order in which they are written is irrelevant, but the order listed here is appropriate, in consideration of temporary file A and the composition of the output contents. There is no necessity to switch the order.

[Subquestion 3]

This question involves sort keys for the news sorting process. Questions regarding the designs of file-processing programs almost always ask about sort keys, so you need to understand how to answer these questions.

Needless to say, when considering sorting keys, you must understand the purpose of the sorting. Normally there are four purposes for sorting, so know these and use them for reference when you consider possible purposes for sorting.

Four Purposes for Sorting

(i) **File matching**: to match between sequentially organized files, each file must be sorted in the same order with respect to the matching key. Sorting is done to prepare for this process.

(ii) **Totaling**: when totals for a particular group, such as for one customer number or for one date, are to be calculated, the data in the same group need to be together. Sorting is done by the key item(s) for particular grouping.

(iii) **Ranking**: to rank multiple data, the data must be ordered from the first place down by the item that is used for ranking. Sorting is done for this purpose.

(iv) **Outputting reports**: In a report output, necessary contents are basically printed out in the order of the input file. Sorting is done to control the order of output data on the report.

In the news sorting process, temporary file B prepared in the news extraction process is sorted first, and then the result (temporary file C) is delivered to the result-output process. Hence, the

reason for this sorting is that the result-output process assumes that the data are already ordered. The contents of the result-output process are to extract, for each category, at a maximum of 10 items from temporary file C and to edit and write the result according to the terminal type. Due to the limit of extracting at a maximum of 10 items, you may think that the purpose is not among the four listed above. However, what is to be extracted is not just any ten random items; there are priorities. According to point (2) of [System Outline], the priorities are "at a maximum of 10 news items backwards by date and time for each registered category", so it corresponds to ranking. For this ranking, the items must be sorted, for each category, from the newest by the date and time of occurrence. Ranking from the newest by date and time means ordering from the largest (i.e., in descending order). Therefore, the answer for blank E is (i) category. For the last two, the correct answers are (h) date of occurrence for blank F and (g) time of occurrence for blank G.

[Subquestion 4]

What is being asked here is the number of records in temporary file C and the number of news items displayed under the given conditions. Having come this far into the question, you have likely understood most of the process contents, so you should be able to respond without reading the question text again. That said, you could probably easily follow the question text to the extent at which we have reviewed the explanation thus far.

From the record we see that the user has registered three categories "politics", "economy", and "sports", and that the user last used the system at 4:00 on April 1, 2008. The current time of usage is 11:00 on April 1, 2008, so it is within 24 hours, and the time range from which records are extracted begins at 4:00 on April 1, 2008.

In temporary file C, among all the news extracted during this time period, only those in the three categories are outputted. The number of these records is, as shown below, 50 (24 + 7 + 19). Hence, blank H is (c).

Time period for extraction →

Category	Number of registered items for each hour											
	00:00 ~ 00:59	01:00 ~ 01:59	02:00 ~ 02:59	03:00 ~ 03:59	04:00 ~ 04:59	05:00 ~ 05:59	06:00 ~ 06:59	07:00 ~ 07:59	08:00 ~ 08:59	09:00 ~ 09:59	10:00 ~ 10:59	
Healthcar	0	0	0	0	0	0	2	0	1	0	0	Total
Education	0	0	0	0	0	1	0	1	0	0	0	
➡ Economy	0	0	0	1	0	2	2	6	8	2	4	24
Entertain	1	0	1	0	1	2	0	1	2	1	1	1
Science	1	0	1	0	1	1	3	2	3	2	5	5
Society	0	1	2	0	1	0	2	3	5	4	3	3
➡ Sports	0	0	0	1	0	0	1	1	0	3	2	7
➡ Politics	0	1	0	0	0	1	4	2	3	6	3	19

Target categories for extraction

On the other hand, for each category the number of news items to be displayed is, at most, 10; so the number of items to be displayed for economy and politics, both with over 10 items, will be 10 each, and, for sports, with fewer than 10 items, will be the number of items: 7. Since blank I is for the total number of displayed items, it will be (a) 27.

Question 2

Q2. Read the following descfription about program design, and then answer the Subquestions 1 through 3.

A program is developed for inventory query as a part of an inventory management system.
This inventory inquiry program handles two types of transactions.

Transaction type "A" is a query about movement(in/out) of parts during a specified period. A parts movement table is the output from this type of transaction.

Transaction type "S" is a query about current inventory. An inventory status table is the output from this type of transaction. If there are any parts in the inventory which are below the minimum stock quantity, then those parts are indicated in red.

Each transaction has the format as shown below. The number of digits, the data type, etc. in each field are checked. In addition, it must have a use authorization number in order to use the inventory-related tables. The existence of the use authorization number of the transaction in the use authorization number table is also checked. Transactions that do not pass these checks are processed as errors.

[Transaction format]

(1) Transaction type "A"

A	Transaction type	Period start (year, month, date)	Period end (year, month, date)

Number of part number	Part number	Part number	. . .	Part number

(2) Transaction type "S"

A	Transaction type

Number of part number	Part number	Part number	. . .	Part number

[Part of database formats related to inventory inquiries]

(1) Parts table

Part number	Part name	Ordering unit	Minimum stock quantity

(2) Inventory table

Part number	Stock quantity

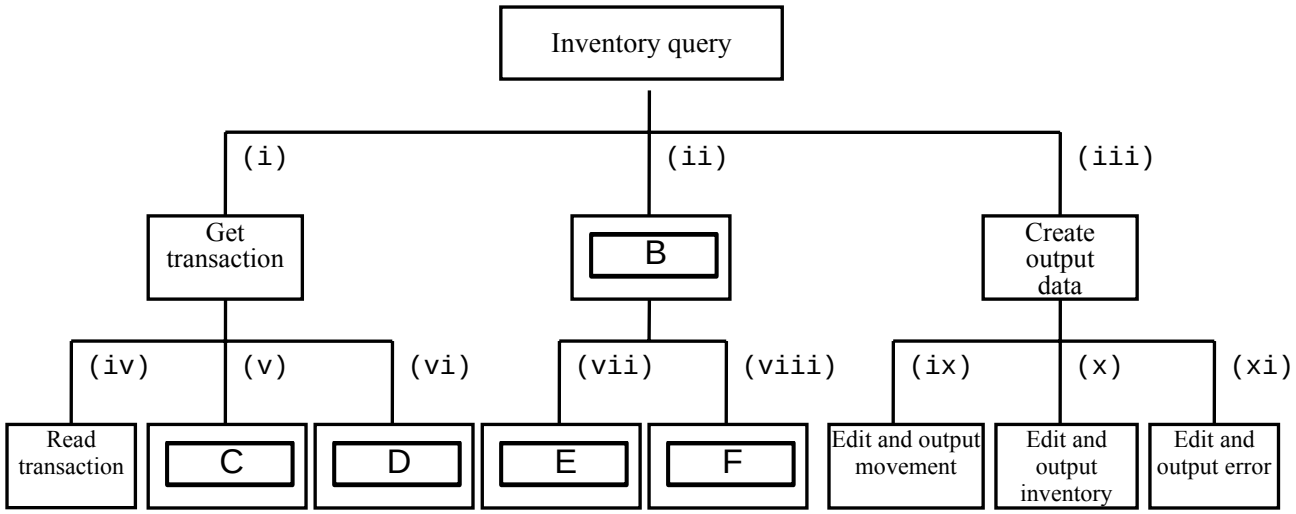
(3) Use authorization number table

Use authorization number

(4) Movement table

Part number	In/Out	Quantity	Time stamp (year, month, date, hour, minute)

[Module structure chart]



[Inter-module interfaces]

Interface number	Input	Output
(i)	–	Transaction, format check flag, use authorization check flag
(ii)	Transaction	Matching records, low inventory flag, part number missing flag
(iii)	Matching records, G, low inventory flag, part number missing flag, format check flag, use authorization check flag	–
(iv)	–	Transaction
(v)	Use authorization number	Use authorization check flag
(vi)	Transaction	Format check flag
(vii)	Transaction	Matching records (parts table, movement table), part number missing flag
(viii)	Transaction	Matching records (parts table, inventory table), low inventory flag, part number missing flag
(ix)	Matching records (parts table, movement table)	–
(x)	Matching records (parts table, inventory table), low inventory flag	–
(xi)	Transaction, H	–

[Flag format]

(1) Format check flag, use authorization check flag

Flag

(Passed: 0, Failed: 1)

(2) Low inventory flag

Number of part number	Flag	Flag	. . .	Flag

(In the order of part numbers in transaction; more than or equal to minimum stock quantity existent: 0, low inventory: 1)

(3) Part number missing flag

Number of part number	Flag	Flag	…	Flag

(In the order of part numbers in transaction; part number existent: 0, part number missing: 1)

Subquestion 1

From the answer group below, select the correct answers to be inserted in the blanks ☐ in the above transaction format.

Answer group for A:

a) "A"
b) "AS"
c) "S"
d) Number of transactions
e) Movement
f) Time stamp (year, month, date, hour, minute)
g) Flag
h) Use authorization number

Subquestion 2

From the answer group below, select the correct answers to be inserted in the blanks ☐ in the above module structure chart.

Answer group for B through F:

a) Get matching records
b) Process records related to inventory status
c) Update inventory table
d) Check transaction format
e) Process records related to movement activities
f) Update the entry and dispatch table
g) Create an order placement message
h) Check use authorization number

Subquestion 3

From the answer group below, select the correct answers to be inserted in the blanks ☐ in the above inter-module interfaces table.

Answer group for G:

a) Transaction
b) Transaction type
c) Number of transactions
d) Time stamp (year, month, date, hour, minute)
e) Part number
f) Number of part number
g) Part name
h) Use authorization number
i) Use authorization number table

Answer group for H:

a) Low inventory flag, part number missing flag
b) Low inventory flag, part number missing flag, format check flag
c) Low inventory flag, part number missing flag, use authorization check flag
d) Low inventory flag, format check flag
e) Low inventory flag, format check flag, use authorization check flag
f) Low inventory flag, use authorization check flag
g) Part number missing flag, format check flag
h) Part number missing flag, format check flag, use authorization check flag
i) Part number missing flag, use authorization check flag
j) Format check flag, use authorization check flag

Answer 2

Inventory Data Inquiry

Correct Answer

[Subquestion 1] A – h

[Subquestion 2] B – a, C – h, D – d, E – e, F – b

[Subquestion 3] G – a, H – h

Comments

This is a question concerning the design of a program that makes queries on inventory data. Subquestions involve transaction data, a module structure chart, and interfaces between modules; all of these have frequently appeared as program design-themed questions on the past exams, so be sure that you understand how to answer these questions.

[Subquestion 1]

A blank is inserted into the transaction formats and you are asked to fill in the blank with its content (item name). The hint is that this blank is mutually set for types "A" and "S". Although the word "transaction" may not be familiar to many examinees, it basically means "input data". For example, in an online program such as this, a "transaction" is the content which is entered on the input screen. Additionally, recall that TR partitioning, a module-partitioning technique frequently appearing on the examinations, is used to divide (partition) process modules for each type when there are some types of the transaction.

Let us now come back to the question at hand and focus our attention on the related part of the question text based on the hint mentioned above. Since this item is common to both types, you should know that it is either to identify the transaction type or to indicate a process that is common to both types of transactions. To identify the transaction type, we already see a category called "transaction type", so this is probably not the correct answer. So let's focus on a process that is in common. Right above [Transaction format], the text indicates that each transaction "must have a use authorization number in order to use the inventory-related tables. The existence of the use authorization number of the transaction in the use authorization number table is also checked". This is carried out regardless of the transaction type, and it is missing from [Transaction format]. Therefore, the correct answer is (h) use authorization number.

[Subquestion 2]

This question is to fill in blanks in the module structure chart. The answer group shows the module names, but there are no detailed explanations concerning the processing, etc., so it may be a little troublesome. To a certain extent, the module names give you an idea, but in parallel blanks such as C and D as well as E and F, you may worry about how the order can be identified.

However, all questions are designed in a way that they can be answered. There must be hints somewhere. Do not be intimidated; look for answers. [Module structure chart] has numbers (i) through (xi). Note that these numbers are also found in [Inter-module interfaces] as interface numbers. So, for instance, the module corresponding to blank C corresponds to the interface number (v), indicating that this receives a "use authorization number" from its upper-class module "Get transaction" and returns a "use authorization check flag". Therefore, you know that blank C must be the module that carries out a process involving checking of use authorization numbers. Looking through the answer group for an appropriate answer, you will find the correct answer (h) "check use authorization number".

Now that you see how to find the correct answers, let us proceed to fill in the rest of the blanks. The module for blank D corresponds to interface number (vi), so the input is "transaction" and the output is "format check flag". Hence, the correct answer is (d) "check transaction format". The other blanks, B, E, and F have a vertical relationship. Often in situations like this, the upper-class module B has a name that covers both modules below it, those corresponding to blanks E and F. So let us take a look at E and F first. As for blank E, the interface number is (vii), so the input is "transaction" and the output is "matching records (parts table, movement table), part number missing flag". From the fact that the matching records contain the parts table as well as the movement table, you can easily imagine that this must be a process corresponding to queries regarding the movement activities of transaction type "A". Looking through the answer group for a choice compatible with this, you will find (e) "process records related to movement activities" and (f) "update the entry and dispatch table". However, recall that the program is about queries of inventory data. Since it is about queries, it is unlikely that updating is the answer. Hence, the correct answer is (e) "process records related to movement activities" for blank E. Since the output for blank F contains "matching records (parts table, inventory table)", you can see that the correct answer is (b) "process records related to inventory status".

Now, the remaining blank is B. This is a module that unifies a process of records related to movement activities and a process of records related to inventory status. There is another hint. We have already used answers (b), (d), (e), and (h) for filling in 4 blanks C, D, E and F. The question text does not say that the answer choices may be used more than once, so we can exclude the choices already selected. Further, (c) update inventory table and (f) update the movement table are both inappropriate for an inquiry program, so they can also be excluded. We are now left with two options: (a) get matching records and (g) create an order placement message. Regarding queries of inventory status, the question text indicates: "If there are any parts in the inventory which are below the minimum stock quantity, then those parts are indicated in red." You can easily imagine that this means "Order these parts." However, in queries of movement activities, this cannot happen. As mentioned earlier, the module of this blank unifies the two modules corresponding to blanks E and F, so (g) "create an order placement message" just doesn't seem right. In contrast, since the output of the interfaces with both modules corresponding to blanks E and F, i.e., return from these modules, contains "matching records" in common, there is no problem if you choose answer (a) "get matching records". Hence, this is the correct answer. In addition, the module to the left of this blank is "get transaction", so the contents of these modules are equivalent to each other, and it is appropriate.

[Subquestion 3]

Finally, we have questions to fill in blanks in [Inter-module interfaces]. If the process of each module is described even briefly, you can think of the contents necessary as input as well as the contents of the output as processing results, but this question is not quite like that. However, do not get frustrated. **There must be hints somewhere.** It is important to calm down and think. Try to recall what kinds of items are necessary as inter-module interfaces. Unnecessary items include those used in one module only. Neither these nor contents that can be obtained through referencing a file or calculating within the module need to be received from upper-level modules. Items needed by processes of a lower-level module are delivered as input from the upper-level module, and items required by an upper-level module are returned as output from the lower-level module. As you can see from the module structure chart, the modules do have a hierarchical structure. For instance, the content called "transaction" is necessary for editing and outputting error in (xi). What obtains this content is "read transaction" (iv). However, between "read transaction" (iv) and "editing and outputting error" (xi), there is no direct interface, so the information cannot be directly delivered. This is obvious. However, often these kinds of points serve as a clue in questions involving inter-module interfaces. Consider how, in this example, the transaction contents are delivered from (iv) to (xi). You will see that the path taken is (iv) read transaction → (ii) obtain transaction → query inventory data → (iii) query output data → (xi) edit and output error message. In other words, to create output data (iii), which is the upper-level module for (ix), (x), and (xi), all items that must be delivered to (ix) through (xi) must be included, except possibly those that are obtained within the module itself and those obtained at a lower-level module. From this point of view, look at the contents that must be delivered to (ix) through (xi) (include all the contents of (ix) through (xi)). They are "matching records, low inventory flag, transaction, and blank H". In contrast, compare these with the input for (iii), and you will see that (a) "transaction" is missing. Hence, this is the correct answer for blank G. At an actual system-developing site, often the transaction may again be obtained at (iii) and (xi), resulting in lower maintainability. However, on the examination questions, you do not need to worry to this extent. Think simply and straightforward.

Now, as for blank H, think in reverse. All items that (iii) receives from its upper-level module, i.e., inventory data inquiry, are either to be used by (iii) itself or to be delivered to lower-level modules (or both, of course). Since the processing of each module is not made clear in this question, let us just assume that all are to be given to lower-level modules for the time being and compare. Then you will see that "part number missing flag, format check flag, and use authorization check flag" are not used. There is another flag—low inventory flag—, but this is used by (x). In inventory status query, if the inventory of a part is low (below some minimum stock quantity), the part is indicated in red, so the low inventory is clearly not handled as an error. On the other hand, you can probably understand that the other three flags indicate some error in the contents of the transaction, requiring an error message to be edited and returned. Hence, the correct answer is (h) part number missing flag, format check flag, use authorization check flag. Actually, there would be no problem even if the low inventory flag were contained here, but there is no choice in the answer group with all four flags.

On the examination, there will be questions on program design and internal design; however, these are not exactly "design" in the true sense of the word. **The question text contains a reason**

for each correct answer so that only one correct answer is possible no matter who answers the question. Also, there are only several patterns of questions. Therefore, study past exam questions to learn the perspective (how to find these reasons for the answers), and then you can assuredly find the correct answers.

8 Program Development

[Scope of Questions]
Programming languages (C, Java™),
Coding, Development environment,
Test methods, etc.

Question 1 C Program

Q1. Read the following description of a C program and the program itself, and then answer Subquestion.

[Program Description]

This is a seat reservation program for a concert hall. The program gets the number of consecutive seats to be reserved as input, then allocates the seats, and returns the result.

(1) The seating arrangement in the concert hall is as shown in the figure below:

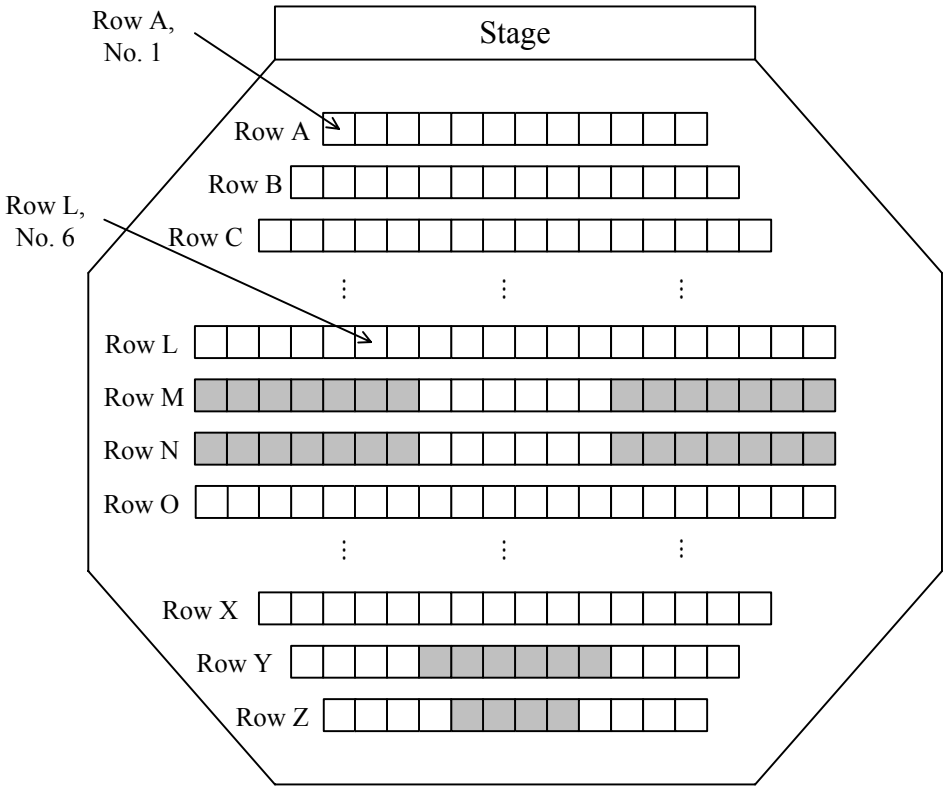


Fig. Seating Arrangement in the Concert Hall

(i) There are 26 rows of seats coded A to Z. The number of seats per row varies from row to row. The number of seats in each row is stored in the global array `cnum` in the order of rows A to Z.

(ii) A seat number consists of a row code and a number. The number indicates the position of the seat starting from the left side in each row when facing the stage. For example, seat "Row L, No. 6" is the 6th seat from the left when facing the stage in the 12th row (row L) up from the stage.

(iii) The seats in the shaded area in the figure are already reserved.

(2) The number of consecutive seats that are to be secured is stored in argument `number` and passed to the program.

(3) The program starts searching for seats starting from the leftmost seat in the first row (row A, No. 1) as seen when facing the stage, seeking the desired number of consecutive seats in the same row, and secures the first available group of consecutive seats it finds. If the seats cannot be secured in row A, it sequentially searches from the leftmost seat in row B, C and so forth until finding the first available group of consecutive seats. If the desired number of consecutive seats cannot be found, it determines that the seats cannot be secured.

(4) If the desired number of consecutive seats are secured , the program returns, as a return value, a pointer to the seat number structure (`SEAT`) which stores the row code where the seats are located, as well as the leftmost seat number (as viewed while facing the stage) among the secured seats. If not found, it returns a null pointer (`NULL`) as the returned value.

(5) The declaration of the `SEAT` structure is as follows.

```
typedef struct {
    char  row;    /* Row code */
    int   no;     /* Number */
} SEAT;
```

(6) The status of each seat in the concert hall is stored in the global variable `status`. If the seat is secured, that seat is marked as reserved.

`Status[i][j]=` { `' '`: Seat is available. / `'R'`: Seat is reserved

The value of the index `i` corresponding to rows A through Z is 0 through 25. The value of the index `j` corresponding to numbers 1 through n is 0 through n−1.

[Program]

```
#define MAXNUM 30    /* Maximum number of seats in row */
#define ROWNUM 26    /* Number of rows */

typedef struct {
    char  row;       /* Row code */
    int   no;        /* Number */
} SEAT;

static char     rname[] = {"ABCDEFGHIJKLMNOPQRSTUVWXYZ"},
                status[ROWNUM][MAXNUM];
static int      cnum[ROWNUM] = {12, 14, 16, 18, 20, 22, 24, 26, 28,30, 30,
30, 30, 30, 30, 30, 28, 26, 24, 22, 20, 18, 16, 14, 12};
static SEAT     empty;

SEAT *book_seat(int);

SEAT *book_seat(int number)
{
    int ridx, cidx, eidx, flg = 0;

    for (ridx = 0; ridx < ROWNUM; ridx++) {
        for (cidx = 0; cidx <= cnum[ridx] - number; cidx++) {
            if (status[ridx][cidx] == ' ') {
                [ A ];
                for (eidx = [ B ]; cidx < eidx; eidx--)
                    if (status[ridx][eidx] == 'R') {
                        flg = 0;
                        break;
                    }
                if (flg == 1) break;
            }
        }
        if (flg == 1) break;
    }

    if (flg == 0)
        return NULL;
    for (eidx = cidx + number - 1; cidx <= eidx; eidx--)
        status[ridx][eidx] = 'R';
    empty.row = [ C ];
    empty.no  = [ D ];
    return &empty;
}
```

Subquestion

From the answer groups below, select the correct answers to be inserted in the blanks ☐ in the above program.

Answer group for A:

a) `cidx++`
b) `cidx--`
c) `cidx += number`
d) `cidx += number - 1`
e) `flg = 0`
f) `flg = 1`

Answer group for B:

a) `0`
b) `cidx`
c) `cidx + 1`
d) `cidx + number`
e) `cidx + number - 1`
f) `number`
g) `number – 1`

Answer group for C:

a) `ridx`
b) `ridx + 1`
c) `ridx - 1`
d) `rname[ridx]`
e) `rname[ridx + 1]`
f) `rname[ridx - 1]`

Answer group for D:

a) `cidx`
b) `cidx + 1`
c) `cidx - 1`
d) `ridx + cidx`
e) `ridx - cidx`

Answer 1

Searching for and Securing Consecutive Unreserved Seats in a Concert Hall

Correct Answer

A – f, B – e, C – d, D – b

Comments

This is a relatively easy question analogous to the process of searching for a designated character string stored in a 2-dimensional array. As you read the program, pay particular attention to the following three points: "role of the array being used", "purposes and contents of the variables being used", and "structure of the processes". You should then be able to fill in the blanks naturally while in the process of reading the program.

In this explanation, we will first try to grasp an overview of the program, organize items to be answered, and then make detailed comments where necessary. The program listing below is the function "book_seat()" written in pseudo language with additional line numbers for explanations and auxiliary lines to help us organize the range and hierarchy of each process block.

```
 1  SEAT *book_seat(int number)
 2  {
 3     ○int ridx, cidx, eidx, flg = 0;
 4
 5     ■for (ridx = 0; ridx < ROWNUM; ridx++) {
 6        ■for (cidx = 0; cidx <= cnum[ridx] - number; cidx++) {
 7           ▲if (status[ridx][cidx] == ' ' ) {
 8                 [    a    ];
 9              ■for (eidx = [    b    ]; cidx < eidx; eidx--)
10                 ▲if (status[ridx][eidx] == 'R') {
11                       flg = 0;
12                       break;
13              ■  ▼}
14              if (flg == 1) break;
15           ▼}
16        ■}
17        if (flg == 1) break;
18     ■}
19     ▲if (flg == 0)
20     ▼   return NULL;
21     ■for (eidx = cidx + number - 1; cidx <= eidx; eidx--)
22     ■    status[ridx][eidx] = 'R' ;
23     empty.row = [    c    ];
24     empty.no  = [    d    ];
25     return &empty;
26  }
```

Have a look through [Program Description] of the question text. Scan the entire [Program] focusing on "array", "variables", and "process structure", and you will obtain the following information:

(Array)

`rname[]`	Row codes A through Z are stored.
`status[][]`	Either ' ' or 'R' is stored to show reservation status.
`cnum[]`	The number of seats for each row is stored. It is used in line 6 in the form `cnum[ridx]`.

(Structure)

`empty.row`	The row code of the designated number of consecutive unreserved seats is stored. Character type.
`empty.no`	The number of the leftmost seat of the designated number of consecutive unreserved seats is stored. Integer type.

(Variable)

`number`	Number of consecutive unreserved seats to be secured; passed as an argument .
`ridx`	Line 7 has "`status[ridx][cidx]`", and line 10 has "`status[ridx][eidx]`", so this corresponds to index `i` to indicate the row in the seating chart.
`cidx`	Line 7 has "`status[ridx][cidx]`", so this corresponds to index `j` to indicate the number in the seating chart.
`eidx`	Since lines 10 and 22 have "`status[ridx][eidx]`", this corresponds to index `j` to indicate the number in the seating chart.
`flg`	Lines 19-20 have "`if (flg == 0) return NULL;`", so `flg` is the flag to indicate whether or not the designated number of consecutive unreserved seats has been found; 0 means not found, and 1 means found.

(Process structure)

Lines 5-18	Block to find consecutive unreserved seats The following three loops are nested. Lines 5-18: `ridx` is used to move down from row to row in the searching process. Lines 6-16: `cidx` is used to check for unreserved seats from left to right. Lines 9-13: `eidx` is used to check for reserved seats from right to left.
Lines 19-20	Block when consecutive unreserved seats were not found. This is clear from line 20: "`return NULL;`".
Lines 21-25	Block when consecutive unreserved seats were found. Lines 21-22: `eidx` is used to change the seat status to "reserved" from right to left. Line 23: Assign row code (`char` type) into `empty.row`. Line 24: Assign number (`int` type) into `empty.no`. Line 25: This returns the address of the variable "`empty`".

In the process of organizing the program in the table above, you should easily notice the following questions and observations concerning the role of the array as well as the roles and functions of the variables.

(1) Role of the array "rname[]"

[Program Description] gives no explanations. Even in [Program], we cannot find any processes using this array. This suggests that one of the blanks must contain this array.

(2) Use of the variable "eidx" and role-separation with the variable "cidx"

The initial value of "eidx" in line 9 is blank B, so at the moment the value is not clear. Related to this, we do not yet know why the two indices "cidx" and "eidx" are both used for numbers. Lines 21 and 22 include a process similar to the pattern of lines 9 and 10, so this should give us a clue.

(3) Use of the variable "flg"

Line 3 initializes this as 0. Later, in line 11, this is set to 0, but there is no process where this is set to 1. Hence, in some blank between lines 3 and 11, this variable must be set to 1.

The following is a summary of what is being asked in each of the blanks:

Blank A: process executed first when "status[ridx][cidx]" is unreserved
Blank B: initial value of the variable "eidx"
Blank C: row code where the designated number of consecutive unreserved seats are found; to be assigned to "empty.row"
Blank D: number of the leftmost seat of the designated number of consecutive unreserved seats; to be assigned to "empty.no"

You should be able to fill in the blanks once you have understood what is known and what is unknown to the extent explained thus far. Next, let us fill in the blanks as we answer the question points.

(1) Role of the array "rname[]" → blank C

To understand the role of the array "rname[] ", we can draw a figure, as shown below, of the mutual relation of the three defined arrays, using it as a hint to identify the number of elements in each and how they are used.

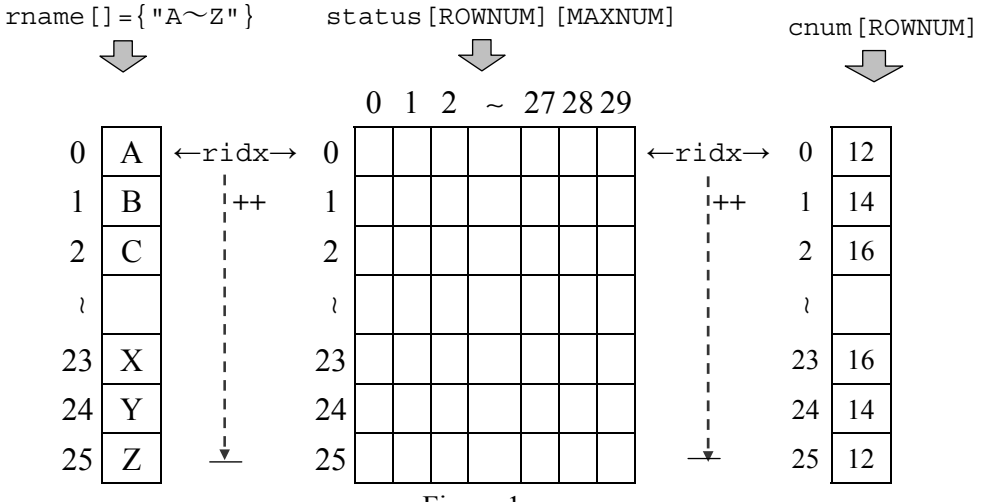


Figure 1

Another clue is that line 6 indicates the number of seats in each row by `cnum[ridx]`. From Figure 1 and `cnum[ridx]`, we see that `rname[ridx]` expresses the row code corresponding to the variable `ridx`. In other words, the array `rname[]` has the function of converting the variable `ridx` into the corresponding row code. The reason that the variable `ridx` needs to be converted to the corresponding row code is that, in line 23, the row code is assigned to `empty.row`. From these observations, we see that what needs to be inserted in blank C is (d) `rname[rdix]`. For your information, arrays such as `rname[]` and `cnum[]` are called conversion tables. You should remember this because it is a technique frequently used.

(2) Use of the variable "`eidx`" and role-separation with the variable "`cidx`" → Blanks B and D

To understand the action of the variable "`eidx`", we confirm its final value, increases and decreases, and usage. In line 9, the variable "`eidx`" decreases until the position next to the one where variable "`cidx`" points to. In the next line, line 10, the program checks whether the seat identified by the variable "`eidx`" is reserved. In other words, the variable "`eidx`" is acting as an index for checking whether the set of seats corresponding to the designated numbers are all unreserved, and its initial value should be the value identifying the rightmost element to be checked. Therefore, the initial value of the variable "`eidx`" should be the sum of the variable "`cidx`" and the designated number of seats given by the variable "`number`", minus 1. Hence, the answer for blank B is (e) `cidx + number - 1`.

The following is a figure confirming the above:

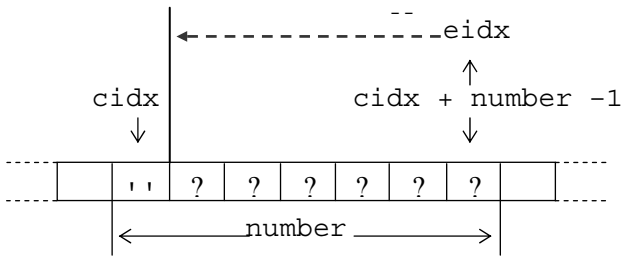


Figure 2

The correct answer for Blank B can also be found using the process, in lines 21 and 22, of changing the designated number of seats to "reserved" starting at the seat identified by the variable "`cidx`". However, these two processes are slightly different in the way they determine consecutive seats. In line 9, since it is known that the seat identified by cidx is unreserved, the variable "`eidx`" needs to be changed only to the seat immediately after the variable "`cidx`" points to. In line 21, on the other hand, the designated number of seats are to be switched to status "reserved", so the variable "`eidx`" needs to be changed to the variable "`cidx`".

Now, let us summarize the roles of the variables "`cidx`" and "`eidx`". We see that the variable "`cidx`" functions to identify the leftmost seat of consecutive unreserved seats while the variable "`eidx`" functions to check whether the designated number of consecutive seats are unreserved. Thus, what needs to be assigned to `empty.no` in line 24 is the number of the seat corresponding to the variable "`cidx`", which is the index that identifies the leftmost seat of the consecutive unreserved seats. To convert index to seat number, we need to add 1 to the index, so blank D should be `cidx + 1`. Thus, the correct answer is (b).

(3) Use of the variable "flg" → Blank A

When we trace the action of the variable "flg", we see that the initial value 0 is set to it in line 3 and the value 0 is again set to it in line 11. Hence, the value 1 needs to be set to it somewhere between lines 3 and 11. Or, you can spot that, in the process of line 14, the value is compared to 1, and conclude that 1 needs to be set to it between lines 3 and 14. Either way, it must be at blank A. In terms of the meaning of these lines, we know that 1 should be substituted immediately after finding the leftmost of the unreserved seats, so we can confirm that this is the appropriate place. Hence, (f) `flg = 1` should be inserted in blank A.

Now that the blanks are filled in, let us check how the program works when it finds the designated number of consecutive unreserved seats. Here, we omit the case when the designated number of consecutive unreserved seats are not found. For your convenience, we have added auxiliary lines using the pseudo language notification to the program listing above so that the program can easily be traced.

Once an unreserved seat is found in line 7, in blank A of line 8, the value 1 is substituted into `flg`. The loop of lines 9~13 checks whether there is any reserved seat within the range identified by the designated number. If not, `flg` remains 1, and the program proceeds to line 14. The condition of line 14 is true, so the program breaks. At the break, the loop of lines 5~16 is skipped, and the program goes to line 17. The condition of line 17 is also true, so the program breaks. At this break, the loop of lines 5~18 is skipped, and the program moves on to line 19. The condition of line 19 is false, so line 20 is skipped, and the program moves to the process of seat reservation in lines 21 and 22.

Hence, we have confirmed that the program is correctly executed when `flg = 1` is inserted into blank A.

This question is categorized as basic, with a low difficulty level. If you have struggled with answering this question, use the comments above to find your own way of reading programs and approach of answering questions. If you have successfully answered this question, use this as an exercise to answer questions with higher levels of difficulty; find several ways that lead you to the correct answers, and organize them.

Question 2 C Program

Q2. Read the following description of a C program and the program itself, and then answer Subquestions 1 and 2.

[Program 1 Description]

This program reads a nonempty source program written in C language from standard input, removes comments, and then outputs it to standard output.

(1) Description on the notation of source programs

(i) "Comments" handled by this program are character strings that start with "/*" and end with "*/", excluding those included in character constants, character string literals, and comments.

(ii) The type of usable characters is as follows.

Space	0	@	P	`	p
!	1	A	Q	a	q
"	2	B	R	b	r
#	3	C	S	c	s
$	4	D	T	d	t
%	5	E	U	e	u
&	6	F	V	f	v
'	7	G	W	g	w
(	8	H	X	h	x
)	9	I	Y	i	y
*	:	J	Z	j	z
+	;	K	[	k	{
,	<	L	\	l	|
-	=	M	]	m	}
.	>	N	^	n	~
/	?	O	_	o	

(iii) Description as shown below is not used.

- Nested comments

  Example `/* aaaaa /* bbbbbb */ ccccc */`

- Three-character notation for graphic characters

  `??= ??( ??' ??< ??> ??) ??! ??-`

(iv) There are no grammatical errors.

(2) Program 1 removes comments in accordance with the following procedure. Since the program simply processes the analyses of character constants, character string literals and comments, they may not be recognized correctly depending on the coding in source programs, resulting in a malfunction.

(i) When detecting a single quote or a double quote, the program interprets it as the beginning of a character constant or character string literal, and then uses function quote to read and output the character string as it is until the program detects the corresponding single quote or double quote.

(ii) When detecting "/*", the program interprets it as the beginning of a comment and skips characters before the first appearance of "*/".

(3) An execution example of the comment removal by Program 1 is shown below.

Input source program

```
/* This program uses fgets to output
 * a line from a file on the screen. */
#include <stdio.h>
int main( void )
{
   FILE *stream;    /* file pointer */
   char line[100];  /* input stream */

   if( (stream = fopen( "crt_fgets.txt", "r" )) != NULL )
   {
      if( fgets( line, 100, stream ) == NULL)
         printf( "fgets error\n" );  /* error message */
      else
         printf( "%s", line);
      fclose( stream );
   }
}
```

Output results after the removal of comments

```
#include <stdio.h>
int main( void )
{
   FILE *stream;
   char line[100];

   if( (stream = fopen( "crt_fgets.txt", "r" )) != NULL )
   {
      if( fgets( line, 100, stream ) == NULL)
         printf( "fgets error\n" );
      else
         printf( "%s", line);
      fclose( stream );
   }
}
```

Fig. Execution Example of the Comment Removal

[Program 1]

```
#include <stdio.h>
void quote( char );

main()
{
    int c1, c2;

    while ( (c1 = getchar()) != EOF ) {
        /* detection of single quote     */
        if ( c1 == '\'' ) quote( '\'' );
        /* detection of double quote     */
        else if ( c1 == '\"' ) quote( '\"' );
        /* detection of slash          */
        else if ( c1 == '/' ) {
            c2 = getchar();
            /* when the next character is an asterisk */
            if ( c2 == '*' ) {
                /* removal of comment character string */
                while ( 1 ) {
                    while ( (c1 = getchar()) != '*' );
                    c2 = getchar();
                    if ( c2 == '/' ) break;
                }
            }
            /* other cases     */
            else {
                putchar(c1);
                putchar(c2);
            }
        }
        else putchar(c1); /* one character read is outputted as it is */
    }
}

void quote( char c )
{   /* extraction of character constant and character string literal    */
    char cc;

    putchar(c);
    while ( (cc = getchar()) != c ) putchar(cc);
    putchar(cc);
}
```

Subquestion 1

From the answer group below, select the code that causes wrong operation when it is entered into program 1.

Answer group:

a)
```
/*   "aaaaaaa"  */
```
b)
```
/* aaa  'a'     */
```
c)
```
if ( c == '\'' ) {
```
d)
```
printf( "  \'     " );
```
e)
```
printf( "aaa  /*  comment  */ \n" );
```

[Program 2 Description]

To solve the problem pointed out in (2) of **[Program 1 Description]**, Program 2 is then written as follows.

(1) The process is divided into three modes: character constant, character string literal, and comment.

(2) An appearance of a single quote switches the "character constant mode" between ON and OFF. However, this does not apply to a piece of code which is an expanded representation using a "\", to a piece of code within a character string literal, or to a piece of code within a comment.

(3) An appearance of a double quote switches the "character string literal mode" between ON and OFF. However, this does not apply to a piece of code which is an expanded representation using a "\", to a piece of code within a character constant, or to a piece of code within a comment.

(4) An appearance of "/*" and "*/" switches the "comment mode" between ON and OFF. However, this does not apply to a piece of code within a character constant or to a piece of code within a character string literal.

[Program 2]

```
#include <stdio.h>
main()
{
    int c1, c2;
    int c_mode = 0;   /* initialize comment mode to off           */
    int quote1 = 0;   /* initialize character constant mode to off    */
    int quote2 = 0;   /* initialize character string literal mode to off */

    for ( c1 = getchar(); ( c2 = getchar()) != EOF; c1 = c2 ) {

        if ( !c_mode ) {       /* when comment mode is off                  */
         /* detecting if \ is in a character constant or a character string literal.*/
            if ( [ A ] && c1 == '\\' ) {
                putchar(c1);
                putchar(c2);
                c2 = getchar();
                continue;
            }
            /* detecting if single quote is not inside a character string literal */
            else if ( !quote2 && c1 == '\'' )
                [ B ];
            /* detecting if double quote is not inside a character constant */
            else if ( !quote1 && c1 == '\"' )
                [ C ];
            /* detecting if / and * are not inside a character constant */
            /* and not inside character string literal */
            else if ( [ D ] && c1 == '/' && c2 == '*' ) {
                [ E ];
                c2 = getchar();
                continue;
            }
            putchar(c1);
        }

        else {
            if ( c1 == '*' && c2 == '/' ) {  /* end of comment? */
                [ E ];
                c2 = getchar();
            }
        }
    }
    putchar(c1);
}
```

Subquestion 2

From the answer groups below, select the correct answers to be inserted into the blanks [ ] in Program 2.

Answer group for A and D:

a) `!quote1`
b) `!quote2`
c) `(!quote1 || !quote2)`
d) `(!quote1 && !quote2)`
e) `(quote1 || quote2)`
f) `(quote1 && quote2)`

Answer group for B, C and E:

a) `c_mode = !c_mode`
b) `c_mode = quote1 && quote2`
c) `quote1 = !quote1`
d) `quote1 = !quote2`
e) `quote1 = quote2`
f) `quote2 = !quote1`
g) `quote2 = !quote2`
h) `quote2 = quote1`

Answer 2

Removing Comments in Source Code in C Language

Correct Answer

[Subquestion 1] c

[Subquestion 2] A – e, B – c, C – g, D – d, E – a

Comments

This is a question concerning the program that reads source code written in C language, removes the comments, and outputs the result. The text of the question takes up a rather large number of pages, but you should be able to answer this question in a relatively short amount of time by using program descriptions and comments within the program. In general, the longer the question is, the more clues it is likely to contain within, so try to answer these questions without worrying about their lengths.

[Subquestion 1]

This is a question of finding a case where a malfunction occurs if the given program is executed. Without reading [Program 1], from the procedures explained under (2) of [Program 1 Description], you can identify the case in which a malfunction occurs. More specifically, we read the following under (2) of [Program 1 Description]: "they may not be recognized correctly … resulting in a malfunction". We can find the answer based on this explanation, as this is followed by the explanation of the procedure to remove comments. Checking how the program operates under each option of the answer group, we see the following:

(a) and (b) fall under (ii), removing everything from "`/*`" to "`*/`" as comments. The operation does not cause any malfunction.

(c) falls under (i) and outputs everything between the first single quote (`'`) and the second one (without any changes) as a character constant. Then, the third single quote is interpreted to begin another character constant, which will be outputted without change. However, in the actual meaning, the third single quote indicates the end of the character constant, so it is not correctly recognized. Consequently, even the "`/*`" appearing in the next line indicating comments is also outputted without deletion, causing a malfunction. As a specific example, in the following case, the comments are outputted.

```
if ( c == '\'' ) {
/* Is "c" a single quotation mark?  */
```

(d) and (e) fall under (i), outputting everything between the first double quote (`"`) and the second one (without any changes) as a character string literal. The operation does not cause any malfunction.

Hence, the correct answer is (c).

Now, we have answered the question without actually reading the program; the answer was obtained based on the program description only. On the actual examination, questions of this type should be answered this way first, and then, if you have extra time, you can read through the program itself to verify that the answer is, indeed, correct.

[Subquestion 2]

The answer groups are divided into those for blanks A and D and those for blanks B, C, and E. There may be some readers who were troubled to see the answer group for A and D. Each of these blanks is within an "if" statement. Generally, a condition is expressed by connecting a variable content with a value using a relational operator , such as in "`c1 == '\\'`". However, none of the options in the answer group for blanks A and D has this format. All of them have only variables or exclamation mark (`!`) coming before the variables, not followed by relational operators.

Condition descriptions in this format are grammatically basic, but the majority of examinees may not have seen this before because the use of this notation is sometimes prohibited by coding rules at development sites, or because identical meaning can be written using relational operators. In C, conditions "`! = 0`" and "`== 0`" can be omitted in description. For example, if a condition formula is "`quote1 != 0`", it can be written "quote1", omitting the relational operator and anything after that. If a condition formula is "`quote1 == 0`", it is equivalent in meaning to "`!quote1`". This is because the data types of C do not contain the logical type; instead, the integer (`int`) type is used, where 0 is false and any non-zero value implies true. So, a relational operator returns some non-zero value if the given formula is true and returns 0 if it is false.

Now, even if you do not know what is stated above, you can figure this out as follows: Note the "`if`" statement two lines above blank A with the comment "when comment mode is off". Its condition is denoted "`!c_mode`". From this, the reader should understand that "`!c_mode`" means "`c_mode == 0`" (because the initial value of c_mode is off, and its value is 0).

In our explanations, we will discuss blanks A and D first and then later deal with blanks B, C, and E. Processes in the blanks of both of these groups are explained in comments, so we need only to convert them to expressions in the program. Technically speaking, any non-zero number can be designated true, but here in our explanations, we will use "1".

Blank A

According to the comments, the "`if`" statement in which blank A is located is detecting if "\" is in a character constant or a character string literal. Here, "detecting the character \" corresponds to "`c1 == '\\'`", so blank A should correspond to "in a character constant or a character string literal". We need to translate it into an expression in C. "in …" suggests that the character constant mode or character string literal mode should be turned ON (1). Translating this in a straightforward way, we get (`quote1 != 0 || quote2 != 0`), but this is not found in the answer group. So, looking again at the answer group, we find (`quote1 || quote2`), which uses the fact that ON (1) and OFF (0) of `quote1` and `quote2` correspond to true (1) and false (0). Hence, the answer for blank A is (e).

Blank D

According to the comments, the "`if`" statement in which blank D is located is detecting if "`/` and `*` are not inside a character constant and not inside character string literal". Here, detecting "/" and "*" corresponds to "`c1 == '/' && c2 == '*'`", so blank D should correspond to "not inside a character constant and character string literal". We need to translate it into an expression in C. "Not…" means the condition that the character constant mode is off (0). Hence, the negation "`!quote1`" is needed. Similarly, this is a condition where the character string literal mode is also off (0), so the negation "`!quote2`" is also needed. The term "and" means we need to connect `!quote1` and `!quote 2` with "`&&`". Hence, the correct answer should be (`!quote1 && !quote2`), which is (d).

Blank B

According to the comments, blank B is the process to be executed when "the comment mode is off" and when it is not true that "`\` is detected either in a character constant or in a character string literal" and when "a single quote is detected but not inside a character string literal". A single quote (`'`) is used to indicate the beginning and the end of a character string constant, so what needs to be done is to switch the character constant mode. More specifically, if it is on (1), it needs to be turned off (0); if it is off (0), it needs to be turned on (1). In other words, the negation of the value of `quote1` is to be set to `quote1`, so we need "`quote1 = !quote1`". The correct answer for B is (c).

If you are not confident in concluding that this is the condition for beginning or ending a character constant, consider the opposite condition, i.e., a situation when a single quote is detected but it indicates neither the beginning nor the end of a character constant. This occurs if "the comment mode is on" or immediately after "the character `\` is found inside a character constant or a character string literal" or if a single quote is found "inside a character string literal". Check the negation of this condition, and you can check the condition of the beginning or ending a character constant.

Blank C

For blank C, use the same line of thinking as for blank B.

According to the comments, what should be inserted into blank C is the process when "the comment mode is off" and when it is not true that "the character `\` is detected either in a character constant or in a character string literal" and when a "double quote is detected but not inside a character constant". A double quote (`"`) is used to indicate the beginning or the end of a character string literal, so what needs to be done here is to switch the character string literal mode. Hence, for blank C, "`quote2 = !quote2`" should be inserted, which is answer (g).

Blank E

Blank E occurs in two places. One is when the "comment mode is off" and when "/ and * are detected not inside a character constant and not inside a character string literal". "/ and *" (i.e., "`/*`") indicate the beginning of a comment, so what needs to be done here is to switch the comment mode from off (0) to on (1). The other occurrence is when it is not true that the "comment mode is off" (meaning when it is on) and at the "end of a comment". Here, the condition of the "`if`" statement is reversed, i.e., to detect "* and /" or "`*/`", which ends a comment. What needs to be done here is to turn the comment mode from on (1) to off (0). A code that can do both of these is "`c_mode = !c_mode`". Hence, the answer for blank E is (a).

(Additional notes)

Finally, although not directly related to the subquestions, we will add a brief remark here concerning three-character notation of graphic characters, which appears under the second bullet point of (iii) under (1) in [Program 1 Description].

Three-character notation for graphic characters is also called tri-graph sequence, and it refers to a set of character strings to denote symbols like "#" and "[" in an environment in which these symbols cannot be used. For instance, "??=" denotes "#", "??(" denotes "[" and "??'" denotes "^".

Question 3 C Program

Q3. Read the following description of a C program and the program itself, and then answer Subquestion.

[Program Description]

Function `execute` draws lines using the marker displayed on the screen .

(1) A bitmap screen has 800 pixels in the horizontal direction and 600 pixels in the vertical direction. Figure 1 shows the screen's coordinate system. A marker (• in Fig. 1) with positional and directional information is shown on the screen. The marker moves in four directions: up, down, left, or right. The locus of the marker is drawn when it is moved.

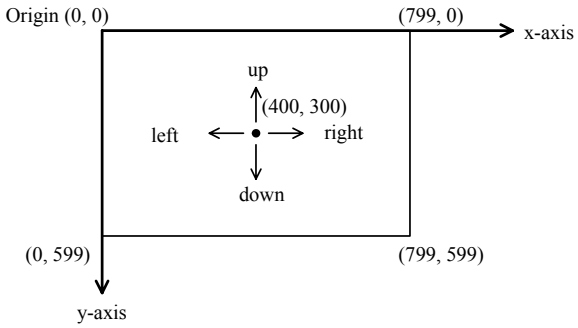


Fig. 1 Screen coordinate system and initial marker state

(2) The marker is represented by `MARKER` type structure `mark`. When the program starts, the marker position is set to (400, 300) and its direction of movement is upward.

```
typedef struct {
    int x;   /* x coordinate of marker */
    int y;   /* y coordinate of marker */
    int dir; /* Marker direction 0:right, 1:up, 2:left, 3:down */
} MARKER;
MARKER mark = {400,  /* Initial x coordinate of marker */
               300,  /* Initial y coordinate of marker */
               1     /* Initial direction of marker (up) */
               };
```

(3) Instructions for operating the marker are defined. Each instruction consists of an instruction code and a value, and is expressed by the structure `INST`.

```
typedef struct {
    char code;   /* Instruction code */
    int  val;    /* value */
} INST;
```

The instructions are stored in the order of execution from the beginning of `insts`, which is an array of the structure `INST`.

(4) The table below lists instruction codes and the descriptions.

Instruction code	Description
{	Repeat instruction execution `val` times from the next instruction to the last one before the instruction code '}' that forms a pair. `val` is an integer greater than 1.
t	Change the direction of movement of the marker by 90° × `val` only in the counterclockwise direction. `val` is a nonnegative integer.
f	Move the marker in the current direction by `val` pixels to draw a line from the source to the destination. `val` is any integer.
}	Indicates the end of a series of instructions to be repeated. `val` is not referenced.
\0	Indicates the end of the instructions. `val` is not referenced.

(5) Figure 3 shows the output of function `execute` executed with the instructions stored in the structure array `insts`, as shown in Fig. 2. Note, however, that the coordinate values in Fig. 3 are added for the sake of explanation and are not actually outputted.

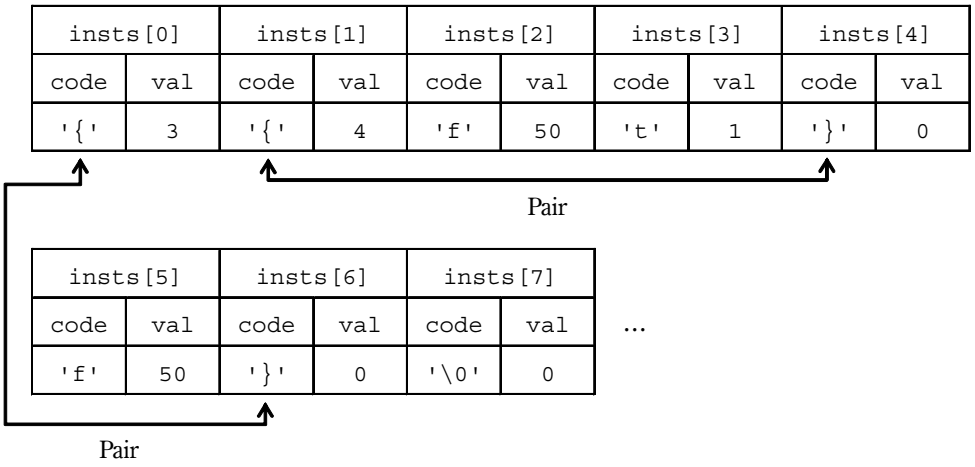


Fig. 2 Example of instructions stored in the structure array `insts`

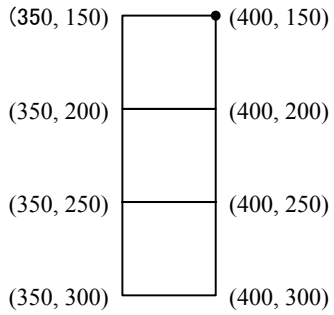


Fig. 3 Results of executing the example in Fig. 2

(6) The following functions are available for drawing lines:

```
void drawLine( int x1, int y1, int x2, int y2 );
```

Features: Draws a line segment connecting coordinate (`x1`, `y1`) and coordinate (`x2`, `y2`) that is in the screen area.

(7) The following functions related to the marker are available:

```
void eraseMarker( MARKER mark );
```

Feature: Do not display the marker when it is in the screen area.

```
void paintMarker( MARKER mark );
```

Feature: Display the marker when it is in the screen area.

(8) Even if the marker has moved out of the screen area and is no longer visible, its position coordinates and direction of movement are retained.

[Program]

```
#define INSTSIZE 100 /* Upper limit of number of instructions */
#define STACKSIZE 50 /* Upper limit of nesting */

typedef struct {
    int x;       /* x coordinate of marker */
    int y;       /* y coordinate of marker */
    int dir;     /* Marker direction  0:right  1:up  2:left  3:down */
} MARKER;

typedef struct {
    char code;   /* Instruction code */
    int  val;    /* value */
} INST;

typedef struct {
    int opno;    /* Element No. of array insts at which the start of loop is defined */
    int rest;    /* Remaining loop count */
} STACK;
```

```
void drawLine( int, int, int, int );
void eraseMarker( MARKER );
void paintMarker( MARKER );

INST insts[INSTSIZE];     /* Structure array for storing instructions */
MARKER mark = {400,       /* Initial x coordinate of marker */
               300,       /* Initial y coordinate of marker */
               1          /* Initial direction of marker (up) */
              };

void execute(){
    STACK stack[STACKSIZE];
    int opno = 0;     /* Element No. of array insts which contains the instruction to be executed */
    int spt = -1;     /* Stack pointer */
    int dx, dy;
    paintMarker( mark );
    while( insts[opno].code != '\0' ){
        switch( insts[stack].code ){
            case '{':
                    stack[ A ].opno = opno;
                    stack[spt].rest = insts[opno].val;
                    break;
            case 't':
                    mark.dir = B ;
                    break;
            case 'f':
                    eraseMarker( mark );
                    dx = ( mark.dir % 2 == 0 ) ? C ;
                    dy = ( mark.dir % 2 == 0 ) ? D ;
                    drawLine( mark.x, mark.y,
                                mark.x + dx, mark.y + dy );
                    mark.x += dx;
                    mark.y += dy;
                    paintMarker( mark );
                    break;
            case '}':
                    if ( stack[spt].rest E ){
                        opno = stack[spt].opno;
                        stack[spt].rest--;
                    } else {
                        F ;
                    }
                    break;
        }
        G ;
    }
}
```

Subquestion

From the answer groups below, select the correct answers to be inserted in the blanks ☐ in the above program.

Answer group for A:

a) `++spt` b) `--spt` c) `spt`
d) `spt++` e) `spt--`

Answer group for B:

a) `( mark.dir + insts[opno].val ) % 2`
b) `( mark.dir + insts[opno].val ) % 3`
c) `( mark.dir + insts[opno].val ) % 4`
d) `mark.dir + insts[opno].val`
e) `mark.dir + insts[opno].val % 2`
f) `mark.dir + insts[opno].val % 3`
g) `mark.dir + insts[opno].val % 4`

Answer group for C and D:

a) `( 1 - mark.dir ) * insts[opno].val : 0`
b) `( 2 - mark.dir ) * insts[opno].val : 0`
c) `( mark.dir - 1 ) * insts[opno].val : 0`
d) `( mark.dir - 2 ) * insts[opno].val : 0`
e) `0 : ( 1 - mark.dir ) * insts[opno].val`
f) `0 : ( 2 - mark.dir ) * insts[opno].val`
g) `0 : ( mark.dir - 1 ) * insts[opno].val`
h) `0 : ( mark.dir - 2 ) * insts[opno].val`
i) `0 : mark.dir * insts[opno].val`
j) `mark.dir * insts[opno].val : 0`

Answer group for E:

a) `< 0` b) `< 1` c) `== 0`
d) `> 0` e) `> 1`

Answer group for F and G:

a) `mark.dir++` b) `mark.dir--` c) `opno++`
d) `opno--` e) `spt++` f) `spt--`

Answer 3

Control of Drawing Marker

Correct Answer

A – a, B – c, C – a, D – h, E – e, F – f, G – c

Comments

This is a question concerning a program that controls a marker that draws pictures in accordance with the instructions stored in a structure array. The processing structure of the program is easy to understand. The functions of the structure unit and variables used, except `dx` and `dy`, are explained in the question text or comments in the program list, so you can focus your attention on the reading of the processes.

In this program, the roles of the data structure and variables used are complicated, as seen in the fact that structure arrays are used. Programs like this should be understood in the following order: organizing the roles of structure arrays and variables → organizing the process structure → understanding the process contents. Further, to understand the process contents, use the sample data shown in the question text. Below, for our explanation, we present the portion of the program list corresponding to the function "`execute()`" with the addition of line numbers, auxiliary lines, block numbers, and comments.

```
 1   void execute(){
 2
 3     STACK stack[STACKSIZE];
 4     int opno = 0; /* Element No. of array insts which contains the instruction to be executed */
 5     int spt = -1; /* stack pointer */
 6     int dx, dy;
 7
 8     paintMarker( mark );
 9
10     while( insts[opno].code != '\0' ) {
11       switch( insts[opno].code ) {
12         case '{':
13                                  stack[ [ a ] ].opno = opno;
14           [1]                    stack[spt].rest = insts[opno].val;
15                                  break;
16         case 't':
17                                  mark.dir = [ b ];
18           [2]                    break;
19         case 'f':
20                                  eraseMarker( mark ) ;
21                                  dx = ( mark.dir % 2 == 0 ) ? [ c ];
22                                  dy = ( mark.dir % 2 == 0 ) ? [ d ];
23                                  drawLine( mark.x, mark.y,
24           [3]                            mark.x + dx, mark.y + dy );
25                                  mark.x += dx;
26                                  mark.y += dy;
27                                  paintMarker( mark );
28                                  break;
29         case ' } ':
30                                  if ( stack[spt].rest [ e ] ){
31                                  opno = stack[spt].opno;
32           [4]                    stack[spt].rest--;
33                                  } else {
34                                          [ f ];
35                                  }
36                                  break;
37       }                                          end switch
38         [ g ] ;
39     }                                            end while
40   }
```

Excerpt from the program list for the function "`execute()`"

(Summary of structure arrays and variables)

As you read through the program list to line 6 with the clues stated in the question text and the comments, you will identify the following facts concerning the roles and contents of structure arrays and variables.

Variable "opno" :the index for "insts", the structure array storing all the instructions. Line 4 initializes this as the value 0. This is an index, i.e., a number to identify an element, so you may anticipate that somewhere there must be an increment process to check the instructions stored in the structure array "insts" in order.

Variable "spt" : the index for the structure array "stack"; line 5 initializes this as the value -1. Details of how this structure array "stack" is used are not yet clear at this stage, but by the comment on the line in which the "stack" type is defined, you can anticipate that this is used to control the number of repetitions. In addition, since the initial value of "spt" is -1, if something is to be stored into the array "stack", then "spt" is probably to be increased before the storing operation. Further, since it is a stack, you can also anticipate that somewhere else there must be a corresponding decreasing process as well.

Structure unit "mark" : elements x and y correspond to the x- and y-coordinates of the marker, so they are used in the drawing process when the instruction code is 'f', and you can anticipate that after drawing, these will be updated to the coordinates of the point to which the marker has moved. The element "dir" is the marker's direction of motion, so again you can expect that this is used in the drawing process when the instruction code is 'f', and will be updated when the instruction code is 't'.

Variables "dx" and "dy" : the roles these variables play are unknown at this stage. From the names, you can expect them to be used in the drawing process when the instruction code is 'f', but we will find out more when we read further.

Figure 1 summarizes all of the above. The structure array "insts" is shown sideways (horizontally) in the question text, but in the figure below we have made it vertical so that the action of the index "opno" is easily displayed.

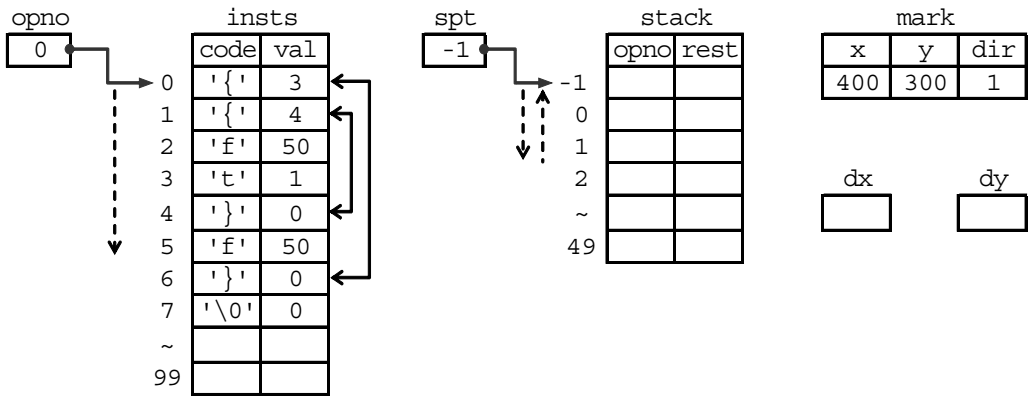


Figure 1: Roles and contents of structure arrays and variables

(Summary of process structure)

The main process of the function "execute" is carried out by the "while-do" loop that continues while the content of "insts[opno].code" is not '\0', which indicates the end. Within this "while-do" loop, there is a "switch-case" block that branches off into 4 cases by the content of "insts[opno].code" and blank G. In the switch-case block, there are four sub-blocks [1][2][3][4], each with a process determined by the instruction code. Blank G is outside the switch-case block, so when every element in an instruction stored in the structure array is

executed, blank G is always executed. Based on this, you may guess that blank G is the process of increasing the index "opno" for the structure array "insts", but let us confirm that later when we read the process contents for understanding.
The processing structure of the function "execute" is easy to understand. However, to reduce errors and misunderstanding, it is advisable that you add auxiliary lines as seen in the program list above or notes for yourself. For instance, come up with a way to clearly show that blank G is outside the switch-case block. Then, you will easily identify the correct answer to be inserted in blank G.

(Reading the process contents)
Having organized structure arrays and variables as well as the process structure, we now go on to fill in the blanks as we read the process contents using the sample data given in the question text. The order in which we read will follow the order in which the sample data are processed. The code in the index 0 of the sample data is '{', so we begin reading from block [1], with blank A.

Blank A:

Blank A is the index for the structure array "stack". As seen from the comment in line 5 concerning the variable "spt" and from the subsequent line to the blank, the index for the structure array "stack" is "spt". But since the initial value of the variable "spt" is -1, we cannot use it as is. Increasing is necessary before checking the value of "opno", an element of "stack". Therefore, blank A should be filled by (a) ++spt.

Figure 2 below shows the state when '{', which is the value of the index 0 in the sample data, is processed through line 14.

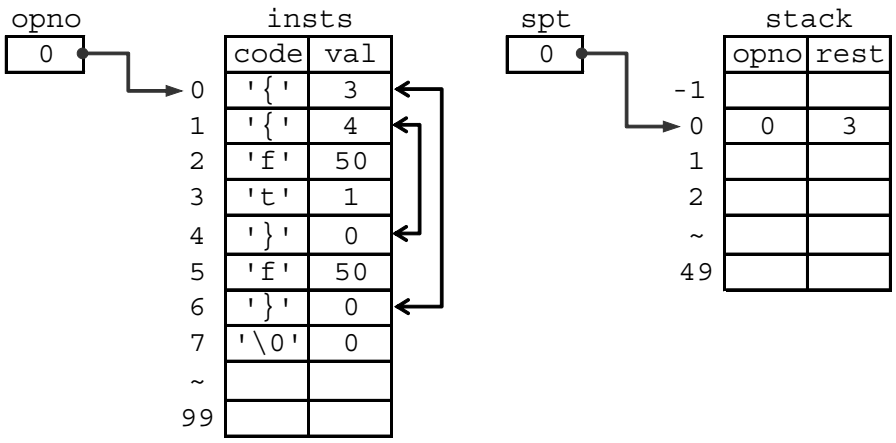


Figure 2: State when '{' with index 0 in the sample data is processed through line 14

After Figure 2, in line 15, the program breaks, skipping the "switch" statement and moving onto line 38, which has blank G.

Blank G:

When we organized the processing structure, we found that blank G is always executed after every instruction stored in the structure array "insts" was processed. We expected that it would be to increase the index "opno". Here, we verify that with the sample data.

Figure 2 is an example of the state immediately preceding the execution of blank G. If nothing is done in blank G and the program continues on to lines 10 and 11, the value of the index "opno" remains 0, so '{' of index number 0, already processed, will be re-processed. However, in actuality, '{' of index 1 should now be processed. In other words, the index "opno" must be increased in blank G. Now, we have confirmed that "opno++" should be inserted into blank G. The answer for blank G is, therefore, (c).

Figure 3 below shows the state when '{' of index 1 is processed through line 38 and the index "opno" is changed to 2.

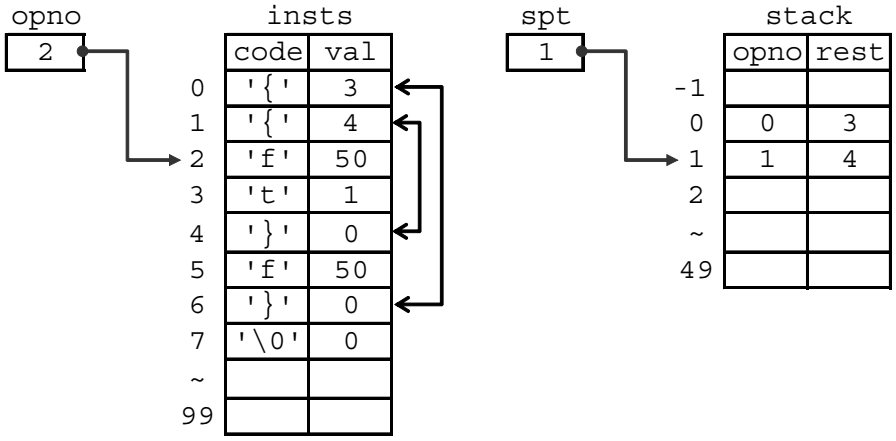


Figure 3: State when '{' of index 1 is processed through line 38 and the index "opno" is set to 2

If the program moves to line 11 in the state shown in Figure 3, since the content of "insts[opno].code" is 'f', the program executes block [3], which has blanks C and D. These lines are similar, so similar reasoning should lead to both answers. Let us first consider C and use that as a reference to fill in D.

Blank C:

In line 21, in which we find this blank, the return value of the formula using the conditional operator "?" is assigned to variable "dx". So let us first examine the role played by this variable "dx".

"dx" is used in lines 24 and 25. In line 24, it is used as an argument for the function "drawLine", which draws a line, as a part of the formula to obtain the x-coordinate of the terminal point. In line 25, this value is added to "mark.x". Because line 27 uses the function "paintMarker" to draw a mark, we understand that this is the addition to update the x-coordinate to the value of the terminal point. Therefore, we now see that variable dx shows the distance (difference) between the x-coordinate of the point before the move (source) and that of the point after the move (destination). In other words, variable dx is the horizontal distance of the motion, pictured below.

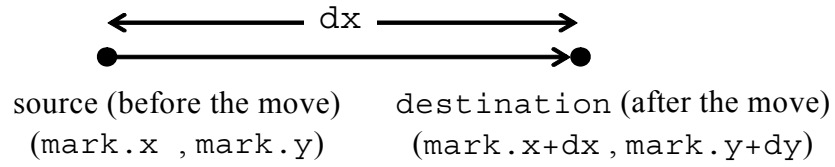


Figure 4: Role of the variable "dx" (Part 1)

However, according to the question text, the distance of motion is stored in "insts[opno].val". This means that the x-coordinate of the destination should be "mark.x + insts[opno].val". As we continue to examine the matter, we identify that when the motion is to the left, we need to subtract the value of "insts[opno].val" from "mark.x". In other words, in that case, the variable "dx" should be replaced by the negative of "insts[opno].val". The following figure illustrates the above:

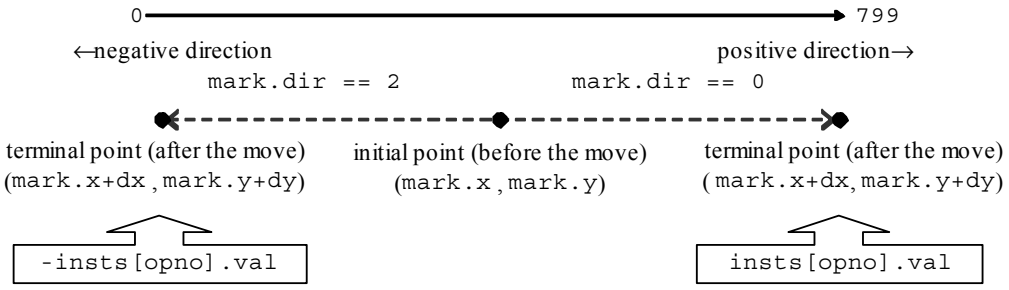


Figure 5: Role of the variable "dx" (Part 2)

Having considered all of the above results, we will find the formula that gives us the value for variable "dx".

The conditional operator "?" is used in the formula to find the value for variable "dx". The conditional operator "?" is used in the form "formula1 ? formula2 : formula3". If formula1 is true, then formula2 is returned; if it is false, then formula3 is returned. Here, formula1 is "mark.dir %2 ==0", which is true only when the value of "mark.dir" is 0 (right) or 2 (left). Therefore, the result of formula2 is assigned to variable "dx" when the marker's direction of motion is right or left, and the result of formula3 when the direction of motion is up or down.

"dx" stores the horizontal distance, so if the motion is up or down, the value should be 0. This narrows down our answer options to those whose formula3 is 0: (a)-(d) and (j). From among these, we are to choose the one such that the result of formula2 is "+insts[opno].val" when the direction of motion is right (mark.dir is 0) and "-insts[opno].val" when the direction of motion is left (mark.dir is 2). This is true with "(1 - mark.dir) * insts[opno].val : 0", which becomes "1 * insts[opno].val" when "mark.dir" is 0 (right) and "-1 * insts[opno].val" when "mark.dir" is 2 (left). Hence, (a) should be inserted into blank C.

This has been a long explanation. Let us summarize. This summary should be useful when considering blank D.

Variable "dx" is the horizontal distance of motion, which is replaced with the positive value of "insts[opno].val" when "mark.dir" is 0 (direction of the motion is right), with the

negative value of "insts[opno].val" when "mark.dir" is 2 (direction of the motion is left), and with 0 when "mark.dir" is 1 or 3 (direction of the motion is up or down). For this, a formula like the following one is appropriate.

```
dx = ( mark.dir % 2 == 0 ) ? (1 – mark.dir) * insts[opno].val : 0
```

Blank D:

Here, we apply the above summary used for blank C.

Variable "dy" is the vertical distance of motion. When "mark.dir" is 0 (right) or 2 (left), it is 0. If "mark.dir" is 1 (up), it is the negative value of "insts[opno].val". When "mark.dir" is 3 (down), it is the positive value of "insts[opno].val".

So the result of formula2 in the conditional operator should be 0. This narrows down the answer options to (e)~(i). Formula3 should be "-1 * insts[opno].val" when "mark.dir" is 1 (up) and "1 * insts[opno].val" when "mark.dir" is 3 (down). Hence, the following equation is appropriate.

```
dy = ( mark.dir % 2 = = 0 ) ? 0 : (mark.dir – 2 ) * insts[opno].val
```

Therefore, (h) should be inserted to blank D.

After 'f' of index 2 is processed in block [3], line 38 increases the variable "opno", and it is changed to 3. Then, the content of "insts[opno].code" becomes 't', so the next block [2] is executed, containing blank B.

Blank B:

According to the question text, if the content of "insts[opno].code" is 't', then the marker's direction of motion is turned counterclockwise by 90 degrees × val. In other words, blank B should have the formula that calculates the value indicating the direction of motion after this turning. Values that indicate directions are limited to integers 0~3, so we need to pick, from the answer group, a formula whose calculation result will be 0~3. Among the choices in the answer group, the only one that satisfies this requirement is (mark.dir + insts[opno].val) % 4. Hence, (c) should be inserted into blank B.

For reference, note that (g) is not correct. For example, if "mark.dir" is 3 and "insts[opno].val" is 5, the calculation will be 3 + 5 % 4 = 3 + 1 = 4, not a valid value for a direction of motion.

We now show Figure 6, which shows the state where 't' of index 3 is processed in block [2] and index "opno" is increased to 4 by line 38.

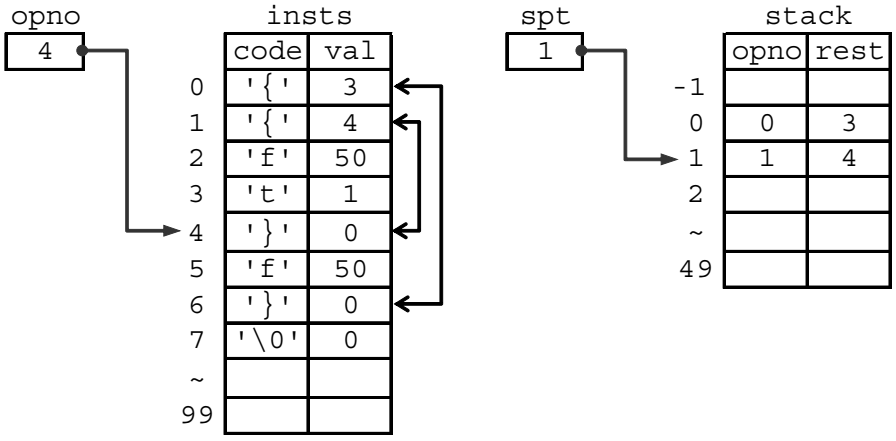


Figure 6: State when index "opno" is increased in line 38 and changed to 4

At this point, the content of "`insts[opno].code`" is '}', so the program now executes block [4], which contains blanks E and F.

Blank E:

Block [4], which contains this blank, is the process when the content of "`insts[opno].code`" is '}', so it can be thought of as the process to repeat the instructions limited by '{' and '}'. This can be verified by the processes in lines 31 and 32 also. For example, in the state shown in Figure 6, if line 31 is executed, the value of the index "`opno`" is updated from 4 to 1, changing to the index of '{' which is paired up with the '}' which was just designated by the index "`opno`". Index "`opno`" is used to extract the instruction stored in the array "`insts`"; it is normally increased by 1 in line 38 each time an instruction is processed. Hence, to update index "`opno`" to a specific value means to branch off to the instruction stored in the location designated by that specific value. In other words, the instructions bounded by "`{`" and "`}`" are again processed (repeatedly). When line 32 is processed, the value of the element "`rest`" (explained as the "number of times yet to be repeated" in the comments) in the structure "`stack`" is decreased.

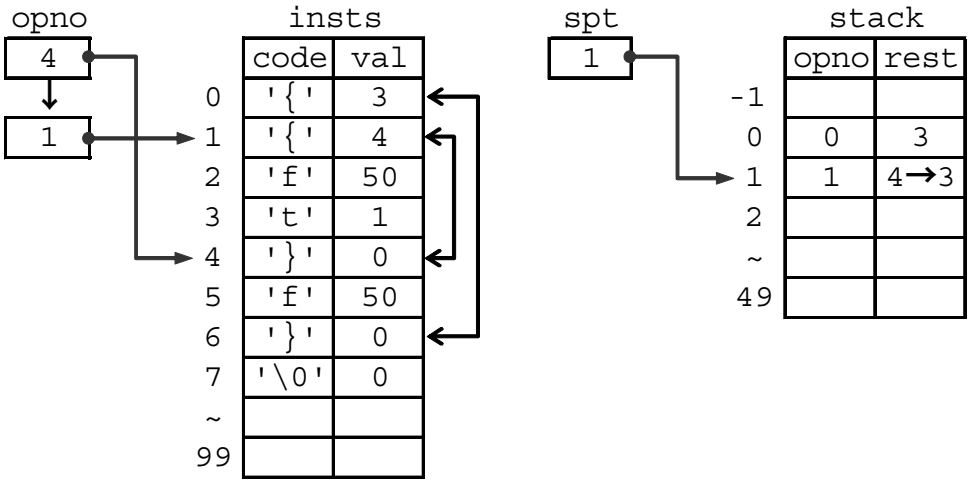


Figure 7: State after line 31 is executed

As seen here, lines 31 and 32 are processes for repetition, so the "`if`" statement in line 30, which includes blank E, must be a condition formula for continuing the repetition. Hence, to fill blank E, you must consider what value of "`stack[stp].rest`" should end the repetition and choose the answer that gives a formula for the opposite condition.

The initial value of "`stack[stp].rest`" is 4, and line 32 ensures that this gets decreased by 1 each time. As we have just mentioned, after this part is executed, the program goes into the repetition that follows. More specifically, if we consider 4 repetitions, the value changes from 4 to 3 when the first execution is finished and the second is about to begin; it changes to 2 when the third is about to begin, and it changes to 1 when the fourth execution is to begin. So, right at the completion of the fourth execution, this value is 1. And at this time, the process for repetition should not be executed; instead, line 34 should be executed.

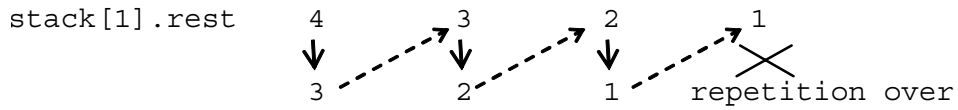


Figure 8

Think backwards now. We need to fill blank G with a conditional formula such that, whenever the value of "`stack[stp].rest`" exceeds 1, the value is true and lines 31 and 32 are processed. Hence, the answer option (e) "`>1`" is to be inserted into blank E.

Blank F:

This is the process to be executed when the content of "`insts[opno].code`" is '}' and "`stack[spt].rest`" is 1, i.e., when the repetition is over. For example, after the instructions limited by '{' of index 1 and '}' of index 4 are repeated 4 times, when blank F in line 34 is to be executed, structure arrays "`insts`" and "`stack`", along with the indices, will be as shown below.

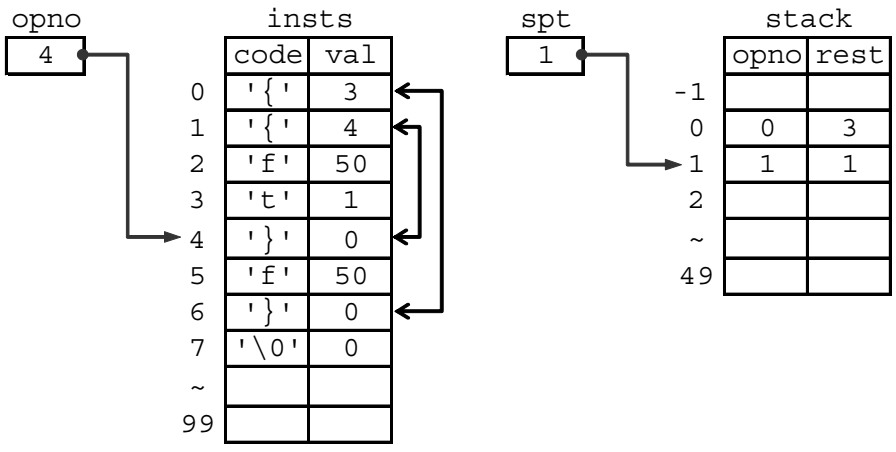


Figure 9: Contents of structure arrays "`insts`" and "`stack`" and their indices

To fill in blank F, identify the problem that can occur if the process continues without performing anything in this blank and then consider how to avoid the problem.

Suppose nothing is done from the state shown in Figure 9 to the next process. First, in line 38, the index "opno" is increased from 4 to 5, causing "insts[opno].code" to be 'f'. Then block [3] is executed. No particular problem occurs in this block. At the completion of block [3], index "opno" is changed from 5 to 6 in line 38. At this point, "insts[opno].code" becomes '}', so the program goes into block [4]. Here, if the value of index "spt" for the structure array "stack" remains 1, a problem occurs. This is because the repetition limited by index 1 and index 4 is already finished, so it must be removed (popped) from the stack. In other words, index "spt" needs to be 0. This can be illustrated in the figure shown below.

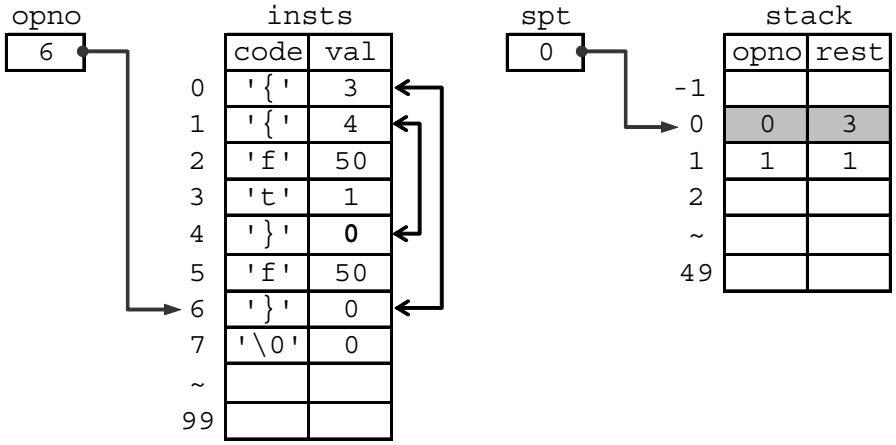


Figure 10: Index "spt"

To create the state shown in Figure 10 when the index "opno" is 6, we needed to decrease the index "spt" after the instructions limited by '{' of index 1 and '}' of index 4 were repeated 4 times (cf. Figure 9). Hence, (f) "spt--" should be inserted in blank F. It is not that we are actually removing the values from the stack, and when '{' is processed next, this is written over in the position of "++spt", so no problem would occur.

Another way to fill blank F is by searching for a correct place to decrease the stack pointer "spt", as there was no process to decrease its value. The timing of decreasing the stack pointer "spt" is right after one "stack[spt].rest" completes its role, so blank F is an appropriate spot for this.

Question 4 Java Program

Q4. Read the following description of a Java program and the program itself, and then answer Subquestion.

[Program Description]

The program consists of the following: a) the class `Encoder`, which converts byte string of binary data according to a certain algorithm and obtains the result as a character string; b) a test class for the class `Encoder`. The conversion algorithm is as follows.

(1) It extracts 6 bits at a time starting from the top of the byte sequence and converts it into a corresponding character in the conversion table. If the number of bytes is not a multiple of 3, then there will be 4 bits or 2 bits of zeros at the end, so that the number of bits will be a multiple of 6.

(2) As many "=" as necessary are added to the end of the converted character string so that the number of characters is the smallest multiple of 4 equal to or greater than 4/3 times of the number of bytes given.

(3) A conversion example is shown in Figure 1 below.

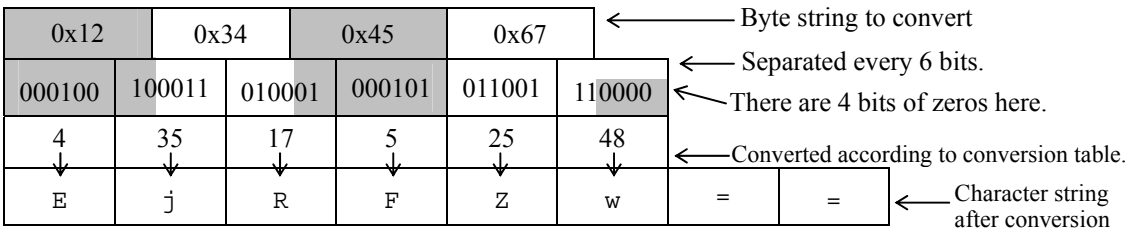


Fig. 1 Conversion Example

(4) The class `Encoder` has two methods: `next` and `hasNext`. The `next` method returns the converted characters one by one in sequence, according to the conversion algorithm. If it is called when there are no characters to return, `java.util.NoSuchElementException` is thrown.

The `hasNext` method is true if a character can be returned without generating an exception the next time that the `next` method is called.

(5) The variable `CHARS` is a table that converts 6-bit numeric values into a single character, and the size of its array is 64. The variable `n` holds information indicating the sequential position of the character to be returned the next time the `next` method is called. (position of the leftmost character is 0 .)

The variable `bin` holds the binary data string to be converted.

(6) The execution result of the test class (`EncoderTest`) for the byte string in Figure 1 is shown in Figure 2.

```
EjRFZw==
```

Fig. 2 Execution Results

[Program]

```
public class Encoder {
    static final char[] CHARS = ("ABCDEFGHIJKLMNOPQRSTUVWXYZ" +
    "abcdefghijklmnopqrstuvwxyz0123456789+/").toCharArray();
    private int n = 0;
    private byte[] bin;
    public Encoder(byte[] bin) {
        if (bin == null) this.bin = new byte[0];
        else this.bin = bin;
    }
    public boolean hasNext() {
        return n < [ A ];
    }
    public char next() {
        char letter;
        int pos = (int)(n * 0.75);
        // Acquires 2 bytes of data containing the 6-bit data targeted for conversion
        if (pos < bin.length) {
            int cell = bin[pos++] << 8;
            if (pos < bin.length) cell += bin[pos] & 255;
            // obtains 6 bits to be converted and converts them into the corresponding character
            letter = CHARS[(cell >> (n + 3) % 4 * 2 + 4) & 63];
        } else {
            if ([ B ])
                throw new java.util.NoSuchElementException();
            else letter = [ C ];
        }
        n++;
        return letter;
    }
}

class EncoderTest {
    public static void main(String[] args) {
        byte[] bin = {0x12, 0x34, 0x45, 0x67};
        Encoder encoder = new Encoder(bin);
        while (encoder.hasNext()) {
            System.out.print(encoder.next());
        }
        System.out.println();
    }
}
```

Subquestion

From the answer groups below, select the correct answers to be inserted in the blanks ☐ in the above program.

Answer group for A:

a) `(bin.length + 2) * 4 / 3`
b) `(bin.length + 2) / 3 * 4`
c) `bin.length * 4 / 3`
d) `bin.length / 3 * 4`

Answer group for B:

a) `!hasNext( )`
b) `hasNext( )`
c) `n < bin.length`
d) `n >= bin.length`

Answer group for C:

a) `' '`
b) `'='`
c) `CHARS[n]`
d) `next( )`

Answer 4

Conversion of Binary Data

Correct Answer

A – b, B – a, C – b

Comments

This is a program that reads byte string of binary data 6 bits at a time and converts them to corresponding character string. Because this expresses 6-bit data with 64 types of characters, this type of conversion is called "BASE64 encoding". Because binary data can be expressed using the characters of the ASCII code, this is used for transmission and reception of binary data mainly in emails. The program consists of one class `Encoder` without any grammatically difficult spots, but there are some complicated concepts, such as the calculation of the number of characters and the fact that byte string is read 6 bits at a time. Also, there are only three blanks, so each of the questions is worth more points; be very cautioned not to make careless errors.

[Program Description]

The constant "`CHARS`" defined in the class `Encoder` is a table for converting bit string of 6 bits obtained from binary data into characters. A character string in Java already shows a reference to a `String` object, and method calling such as "`toCharArray`" is possible, as seen in this program.

In the class `Encoder`, the program receives byte strings in binary data with a constructor and obtains one character at a time by the method "`next`". The "`hasNext`" method is the method that returns, by `boolean`, whether there are additional character data remain.

First, let us look at the process of converting binary data into characters, carried out using the "`next`" method, although it is not always necessary to understand the content we explain here in order to answer the questions.

As the question text says, the variable "`n`" retains the character position to be returned next (0 for the leading character). The array "`bin`" of binary data stores 8-bit data as each of its elements, splits them into 6 bits each, and converts them to characters for output, so the number of characters that is output does not correspond precisely to the element number (index) of the array "`bin`". Hence, by multiplying "`n`" by 0.75 (= 6 / 8), we obtain the index of the element of the array "`bin`" corresponding to "`n`", and this is stored in the variable "`pos`". Here, by casting with `(int)`, we truncate the calculation result after the decimal point.

The problem presented here is that in the 8-bit data corresponding to the index obtained in the variable "`pos`", the necessary 6-bit data may not be always contained. For example, the 6 bits corresponding to the first character are clearly the 6 leading bits with index 0. How about the data corresponding to the next character (`n = 1`)? Of course, they are the 6-bit data that follow, but out of the 8 bits with index number 0, there are only 2 bits left. Hence, we also need the leading 4 bits of index 1 as well. On the other hand, when `n = 1`, the value of the variable "`pos`" is the integer part (truncating after the decimal number) of $1 \times 0.75 = 0.75$, which is 0. Hence, the value of "`pos`" obtained from the variable "`n`" is actually the index of the element that contains the leading bit of

the 6-bit string, and the six necessary bits may not be entirely contained in the data expressed by the index. Therefore, if the value of "pos" is less than "bin.length", the following process is used to obtain the possibly necessary 16-bit data into "cell" from the array "bin".

```
int cell = bin[pos++] << 8;
if (pos < bin.length)  cell += bin[pos] & 255;
```

Here, concerning the process "bin[pos++] << 8", since "bin" is a byte type (8-bit) array and since 8-bit left shifting is done to the 8-bit data, you may think that the result will be 0. However, in shift operations in Java, even if the variable is of the byte type or the short type (16-bit), all variables are first converted to the int type (32-bit) and then the operation is carried out (if the result is stored into a space with fewer than 32 bits, the higher-order bits are truncated). If there are more data after that (pos < bin.length), then the 8 bits of the next data are added (cell += bin[pos] & 255) so that 16-bit data can be stored in the variable "cell". The reason the bitwise operation "AND" is performed with 255 (1111 1111 in binary) before the addition is simply to add only the lowest 8 bits of "bin [pos]" (in our case, "bin[pos]" originally has only 8 bits, so this is not necessary, but normally this type of bit operation is carried out). So, after the 16-bit data that may correspond to the character position identified by the variable "n" are stored in "cell", we move to the following:

```
letter = CHARS[(cell >> (n + 3) % 4 * 2 + 4) & 63];
```

Note the part "(cell >> (n + 3) % 4 * 2 + 4) & 63" in this line. By this calculation we obtain the 6 bits that will be converted, and that becomes the index of the array "CHARS", and the corresponding character is found.

Consider this formula "(cell >> (n + 3) % 4 * 2 + 4) & 63" again. Normally, the calculation is divided up into 4 cases depending on the value of n, but here it seems that the 4 cases are combined into one. For example, consider the binary data below. Note that the part "& 63" is to obtain the lowest-order 6 bits by taking the "&" with $63_{10}=111111_2$.

	[0]	[1]	[2]	[3]	
Data in array "bin"	0x12	0x34	0x45	0x67	……
(Binary number)	00010010	00110100	01000101	01100111	……

Three 8-bit data give us exactly four 6-bit data, so the first three 8-bit data in bin[0], bin[1], and bin[2]

pos =	0	1	2
	00010010	00110100	01000101

give us the following four 6-bit data (characters):

n =	0	1	2	3
	000100	100011	010001	000101

[1] When n＝0

"cell" contains the bit string 000100|10　00110100 concatenating bin[0] and bin[1]. To obtain the 6 bits in the box, we shift this string 10 bits to the right and take the lowest 6 bits.

(0 + 3) % 4 * 2 + 4 = 3 * 2 + 4 = 10

[2] When n=1

"cell" contains the bit string 000100|10　0011|0100 concatenating bin[0] and bin[1]. To obtain the 6 bits in the box, we shift this string 4 bits to the right and take the lowest 6 bits.

(1 + 3) % 4 * 2 + 4 = 0 * 2 + 4 = 4

[3] When n=2

Now, pos = 1, so "cell" contains the bit string 0011|0100　01|000101 concatenating bin[1] and bin[2]. To obtain the 6 bits in the box, we shift this string 6 bits to the right and take the lowest 6 bits.

(2 + 3) % 4 * 2 + 4 = 1 * 2 + 4 = 6

[4] When n=3

Since pos = 2, "cell" contains the bit string 01|000101|　01100111 concatenating bin[2] and bin[3]. To obtain the 6 bits in the box, we shift this string 8 bits to the right and take the lowest 6 bits.

(3 + 3) % 4 * 2 + 4 = 2 * 2 + 4 = 8

For n=4 and larger, the four processes above (n=0-3([1]~[4])) are repeated, so the conditions n=0~3 can be replaced by n % 4 = 0~3. Thus, we can summarize the number of bits by which the variable "cell" is shifted to the right as follows:

n % 4	Bits by which "cell" is shifted to the right
0	10
1	4
2	6
3	8

Changing the form so that (+4) appears in the formula, we get this:

n % 4	Bits by which "cell" is shifted to the right
0	6 + 4
1	0 + 4
2	2 + 4
3	4 + 4

Factoring out 2 from the first term, we get this:

n % 4	Bits by which "cell" is shifted to the right
0	3 * 2 + 4
1	0 * 2 + 4
2	1 * 2 + 4
3	2 * 2 + 4

Hence, the number of bits by which "cell" needs to be shifted can be obtained by the formula "(n + 3) % 4 * 2 + 4". Once "cell" is shifted to the right by this appropriate number of bits, we do the operation bitwise AND on the result and 63. Then, we can obtain the 6 bits we need.

[Subquestion]

Blank A

Consider the condition under which "`hasNext`" returns true. The question text provides the following description: "as many '=' as necessary are added to the converted character string so that the number of characters is the smallest multiple of 4 equal to or greater than 4/3 times of the number of bytes given". Hence, while this is true, "=" continues to be outputted even if there is no more binary data that can be converted to a character. Then, the class "`EncodeTest`" outputs characters while "hasNext" is true, so as long as the number of characters obtained by the "next" method is less than the "smallest multiple of 4 equal to or greater than 4/3 times of the number of bytes given", "`hasNext`" must return true. On the other hand, the variable "`n`" appearing before blank A is showing the number of characters already obtained by the "`next`" method. Hence, for blank A we need a formula that calculates the "smallest multiple of 4 equal to or greater than 4/3 times of the number of bytes given". You must be careful that, in Java, division of an integer by an integer truncates the decimal (fractional) part, giving only an integer result. Find the answer with this in mind. The correct answer is (b) "`(bin.length + 2) /3 * 4`".

The reason why add 2 to "`bin.length`" before dividing it by 3 is so that the quotient is exactly `bin.length / 3` if "`bin.length`" is a multiple of 3 and the quotient is "`bin.length / 3 + 1`" otherwise. Multiplying this quotient by 4 correctly gives us the "smallest multiple of 4 equal to or greater than 4/3 times".

As for the other options, (a) and (c) do the multiplication by 4 before division by 3, so the result may not be a multiple of 4. (d) is incorrect since its result is less than 4/3 of the number of bytes given.

Blank B

Here we are to find a condition under which, in the "`next`" method, the exception `java.util.NoSuchElementException` is thrown . Related to this exception is point (4) of [Program Description] in the question text. It says, "if it is called when there are no characters to return…", the exception corresponds to this condition. When there are binary data to be converted (`pos < bin.length`), the "`next`" method converts the data to a character and returns the character obtained. Even if it cannot convert the data, it must continue to output "=" until it reaches the required number of characters because the question text requires as follows: "as many '=' as necessary are added to the converted character string so that the number of characters is …equal to or greater than 4/3 times of the number of bytes given". Searching through the answer group, there is no answer that carries out such calculation. However, by comments for blank A above, "`hasNext`" returns a `boolean` value of whether the required number of characters is reached, through the result of calculating the number of characters. So we use this. In other words, we use the fact that there are characters to be returned while `hasNext` is true. Blank B is the condition to throw the exception, so we take the negation of "`hasNext`"; the correct answer is (a) "`!hasNext()`".

Blank C

As explained for blanks A and B, the "`next`" method continues to output "=" until it reaches the required number of characters, even after the byte data to be converted have run out. Blank C is the output for this purpose. Therefore, the correct answer is (b) '='.

Question 5 Java Program

Q5. Read the following description of a Java program and the program itself, and then answer Subquestion.

[Program Description]

Interface `CharIterator` defines an operation to extract characters (`char`) from its instance in order, independent of the data structure. In `CharIterator`, the following methods are defined:

`public char next()`

If the next character exists, it is returned. If it does not exist, `java.util.NoSuchElementException` is thrown. If, for example, the `CharIterator` has *n* characters, the first character is returned for the first call, and the second character is returned for the second call. In this way, characters are returned in order until the *n*-th call is made. For the *n*+1-th or a subsequent call, `NoSuchElementException` is thrown.

Note that, `NoSuchElementException` is a subclass of `java.lang.RuntimeException`.

`public boolean hasNext()`

If the next character exists, `true` is returned. If it does not exist, `false` is returned.

Class `CharIteratorFactory` defines a method that returns `CharIterator` that matches the data type specified as the argument.

In `CharIteratorFactory`, the following class methods are defined:

`public static CharIterator getCharIterator(String data)`

`CharIterator`, which extracts characters in order from the `String` specified as an argument, is returned. If a `null` argument is specified, `NullPointerException` is thrown.

`public static CharIterator getCharIterator(char[][] data)`

`CharIterator`, which extracts characters in order from the array of arrays (two dimensional character array) whose elements' type is `char` specified as the argument, is returned. If a `null` argument is specified, `NullPointerException` is thrown.

Characters are to be extracted in ascending order of index values of character array and it's array . That is, for m, whose type is `char[][]`, their elements `m[i][j]` are extracted in ascending order of `i`, and for the same '`i`', in ascending order of `j`.

Class CharIteratorTest is a program that tests methods defined in CharIteratorFactory. The execution result of method main is shown in the figure below:

```
'H' '1' '6'
'2' '0' '0' '4'
```

Fig. Execution Results of CharIteratorTest.main

[Program 1]

```
public interface CharIterator {
    public boolean hasNext();
    public char next();
}
```

[Program 2]

```
import java.util.NoSuchElementException;

public class CharIteratorFactory {
    public static CharIterator getCharIterator(String data) {
        if (data == null)
            throw new NullPointerException();
        // Generates and returns an instance of CharIterator that
        // returns characters in order from String data.
        return [          A          ];
    }

    public static CharIterator getCharIterator(char[][] data) {
        if (data == null)
            throw new NullPointerException();
        // Generates and returns an instance of CharIterator that
        // returns characters in order from two dimensional character array.
        return [          B          ];
    }
}
```

```
class StringCharIterator implements CharIterator {
    private String data;
    private int index = 0;

    StringCharIterator(String data) {
        this.data = data;
    }
    // Check whether or not the next characters exist in data.
    public boolean hasNext() {
        return [          C          ];
    }

    public char next() {
        // If the next characters do not exist, throw NoSuchElementException.
        if (index >= data.length())
            throw new NoSuchElementException();
        // Returns the character next to data and updates index value.
        return [          D          ];
    }
}


class Char2DArrayCharIterator implements CharIterator {
    private char[][] data;
    private int index1 = 0, index2 = 0;

    Char2DArrayCharIterator(char[][] data) {
        this.data = data;
    }
    public boolean hasNext() {
        // If an element of data[index1][index2] exists, true is returned.
        // If it does not exist, the next element defined is searched.
        // If the next element does not exist, false is returned.
        for (; index1 < data.length; index1++) {
            if (data[index1] != null
                    && index2 < data[index1].length) {
                return true;
            }
            [          E          ];
        }
        return false;
    }
    public char next() {
        // Calls method hasNext to check for next element. If an element exists,
        // the element is returned, and the index value is updated.
        // If no element exists, NoSuchElementException is thrown.
        if (hasNext()) {
            return [          F          ];
        }
        throw new NoSuchElementException();
    }
}
```

[Program 3]

```
public class CharIteratorTest {
    public static void main(String[] args) {
        CharIterator itr =
            CharIteratorFactory.getCharIterator("H16");
        printIterator(itr);

        itr = CharIteratorFactory.getCharIterator(
                new char[][] {   { '2' },
                                 { '0' },
                                 null,
                                 { '0', '4' }});
        printIterator(itr);
    }
    private static void printIterator(CharIterator itr) {
        while (itr.hasNext()) {
            System.out.print("'" + itr.next() + "' ");
        }
        System.out.println();
    }
}
```

Subquestion

From the answer groups below, select the correct answers to be inserted in the blanks ☐ in the above program.

Answer group for A and B:

a) `getCharIterator((char[][]) data)`

b) `getCharIterator((String) data)`

c) `new Char2DArrayCharIterator()`

d) `new Char2DArrayCharIterator(data)`

e) `new StringCharIterator()`

f) `new StringCharIterator(data)`

Answer group for C:

a) `index < data.length()`

b) `index <= data.length()`

c) `index >= data.length()`

d) `index++ < data.length()`

e) `index++ <= data.length()`

f) `index++ >= data.length()`

Answer group for D:

a) `data.charAt(++index)`
b) `data.charAt(--index)`
c) `data.charAt(index + 1)`
d) `data.charAt(index++)`
e) `data.charAt(index--)`
f) `data.charAt(index)`

Answer group for E:

a) `index1 = 0`
b) `index1 = data.length`
c) `index1 = index2`
d) `index2 = 0`
e) `index2 = data.length`
f) `index2 = index1`

Answer group for F:

a) `data[index1++][index2++]`
b) `data[index1++][index2]`
c) `data[index1][++index2]`
d) `data[index1][--index2]`
e) `data[index1][index2 + 1]`
f) `data[index1][index2++]`

Answer 5

Program That Lists Characters

Correct Answer

A – f, B – d, C – a, D – d, E – d, F – f

Comments

This is a program that receives data of a 2-dimensional array containing character strings (`String` -type) or characters, extracts them one character at a time and then displays them. This program contains "`hasNext`" and "`next`", which should be quick and simple to answer. Attention must be paid to the change in the index of the array inside `Char2DArrayCharIterator`, which uses a 2-dimensional array, but this is the only point that may be difficult in terms of algorithm. Overall, this should be an easy question.

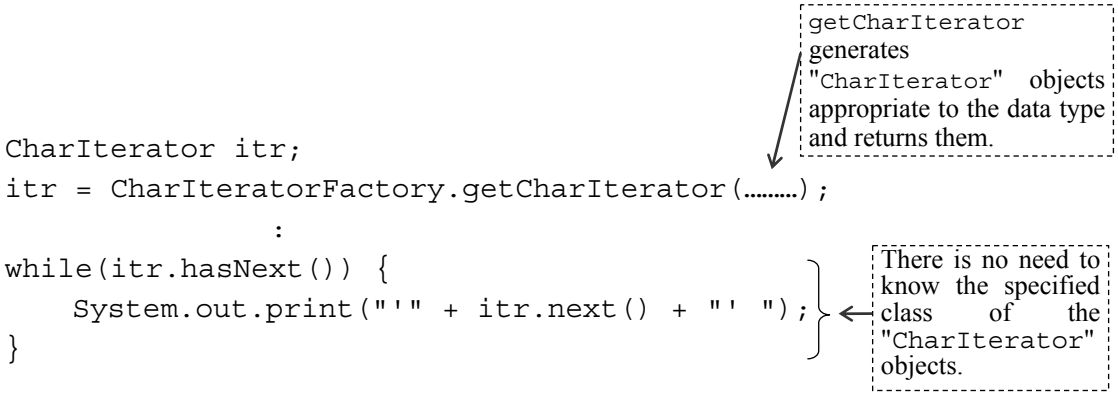
[Program Description]

First, in [Program 1], the "`CharIterator`" interface is defined. Then, in [Program 2], classes "`CharIteratorFactory`", "`StringCharIterator`", and "`Char2DArrayCharIterator`" are defined.

"`CharIteratorFactory`" is a class declared as `public`, but the other two classes are not `public` classes. These classes that are not `public` can be used only in the same package; they cannot be used directly from other packages (this program is not declared as a `package`, but even in this case it forms a package called a "unnamed package"). However, the objects generated by these classes can be method-called by other packages through `CharIterator`-type object variables. In such cases, methods overridden in the `StringCharIterator` or the `Char2DArrayCharIterator` class are called. In other words, even without declaring specified classes like these two "`public`", the functions defined in these classes can be used through the `CharIteratorFactory` class (object generation) and the `CharIterator` interface (method use), so the specified classes themselves are intentionally not made public.

In Java programming, we often hide some specified classes on purpose while using the functions of these classes indirectly. This is to weaken the coupling between classes (loose coupling) and to enhance the maintainability and extensionality of the program.

In the program at hand, the `static` method "`getCharIterator`" is overloaded (multiply-defined) in the class "`CharIteratorFactory`" so that objects appropriate to their data types are generated and returned. Further, the program is designed so that the class "`CharIteratorTest`", which uses these objects in [Program 3], can operate without knowing the existence of classes like `StringCharIterator` and `Char2DArrayCharIterator`.



Blanks A and B

These are questions regarding the return values of the two `static` methods "`getCharIterator`" overloaded in the "`CharIteratorFactory`" class. The return value for each of these two methods is of the "`CharIterator`" type, so an object of the class that implements the "`CharIterator`" interface must be generated and returned (objects cannot be generated from the "`CharIterator`" interface). As for the class of objects to be generated, we can think of two: the class "`StringCharIterator`" and the class "`Char2DArrayCharIterator`". Looking at the constructor of each of these classes, we find that the constructor of the class "`StringCharIterator`" receives "`String`" objects as arguments while the constructor of the class "`Char2DArrayCharIterator`" receives `char`-type 2-dimensional arrays.

For these reasons, the only choice for generating objects and returning them correctly is the combination of (f) "`new StringCharIterator(data)`" for blank A and (d) "`new Char2DArrayCharIterator(data)`" for blank B. The variable "`data`" in blank A is of the `String` type, and the variable "`data`" in blank B is a 2-dimensional array of the `char` type (`char[][]`), so these must be the correct answers by data type.

As for the other answer options, because "`String`" cannot be cast to "`char[][]`" and "`char[][]`" cannot be cast to "`String`", neither (a) nor (b) can be correct, as one "`getCharIterator`" cannot call the other. Besides, it is meaningless to recursively call themselves (calls continue endlessly, eventually resulting in an error). As for (c) and (e), they are wrong because a constructor that does not take arguments is not defined in the classes "`StringCharIterator`" or "`Char2DArrayCharIterator`".

Blank C

This is a question concerning the "`hasNext`" method in the class "`StringCharIterator`". In this class, `String` objects are received with a constructor and are stored in the `private` field "`data`", from which characters are taken and returned one at a time. There is another field "`index`" in this "`StringCharIterator`" class, and its initial value is defined as 0. We can guess that this points to the current character position in the character string (the leading position of a character string is 0, and the end position is "`data.length( ) - 1`"). As written in [Program Description], the "`hasNext`" method is a method that returns the "existence of a next character" by `boolean (true/false)`. Therefore, we need to set it up so that "`true`" is returned if this `index` is smaller than the length of the character string "`data.length()`" and "`false`" is returned if the `index` is equal to or greater than `data.length()`. Therefore, the

answer is (a) "`index < data.length()`".

For your information, the option (d) "`index++ < data.length()`" includes the incrementing process also, and, depending on the answer for blank D, this answer could enable the program "`CharIteratorTest`" to operate properly (if blank D is "`data.charAt(index - 1)`", which is not found in the answer group). However, even in that case, "`index`" should not be incremented in this "`hasNext`". The value of "`index`" should be incremented only after one character is obtained, and "`hasNext`" must return the same result no matter how many times it is called until "`next`" is called again. Further, when "`hasNext`" is called multiple times, there should not be any character that gets left out and does not get extracted by "`next`". These points are not directly relevant in answering the question at hand, but in actual programming, when you create "`Iterator`", you should keep these things in mind.

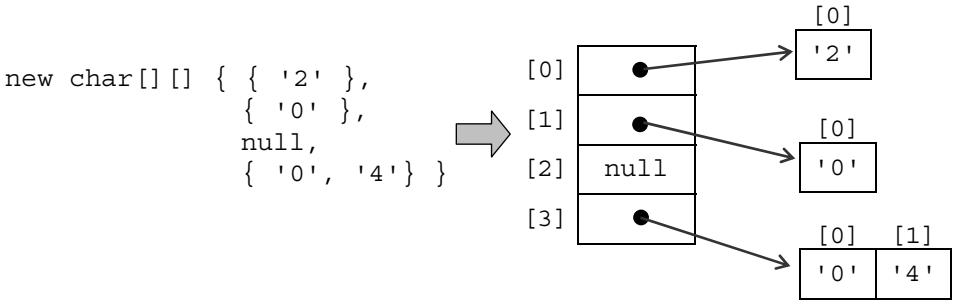
Blank D

This question is about the "`next`" method in the class "`StringCharIterator`". Concerning this method, [Program Description] states, "If the next character exists, it is returned. If it does not exist, …" So the function of this method is to extract and return, in order, one character at a time from the character string received as the argument. For this, as seen in the comments for blank C above, blank D must contain a process of obtaining the character in the position designated by "`index`" from the character string "`data`" and then a process to increment the value of "`index`" after that. Hence, the correct answer is (d) "`data.charAt(index++)`".

Blank E

This question is about the method "`hasNext`" in the class "`Char2DArrayCharIterator`". In this class, a constructor is used to obtain a 2-dimensional character array, which is then stored in a `private` field "`data`". Two other `private` fields, "`index1`" and "`index2`", are defined with initial value 0. From the comments in the "`hasNext`" method and from the answer group for blank F, these are easily recognized as the first and second indices of the 2-dimensional array "`data`".

A 2-dimensional array in Java has a structure referred to as an "array of arrays". The 2-dimensional array delivered from the "`main`" method of the "`CharIteratorTest`" class is structured as shown here:



In this array, `data[2]` stores `null` while `data[0]`, `data[1]`, and `data[3]` contain a reference to each array domain. The sizes of these domains are, respectively, `data[0].length`(=1), `data[1].length`(=1), and `data[3].length`(=2). Furthermore, by "`data.length`", one can obtain the size of "`data`" as an "array of arrays" (=4). The main point of the question here is how to handle this group of arrays in which the sizes are different

and sometimes the content could be null.

In the "hasNext" method, "for" statement is used. A casual look may cause you to think that a repetitive process is carried out, but the program has the following "if" statement.

```
if(data[index1] != null
  && index2 < data[index1].length) {
  return true;
}
```

This is designed so that the program immediately escapes this method with "return true;" whenever the condition "data[index1] != null && index2 < data[index1].length" holds. Hence, this "for" statement may actually be reached once during the execution and the program may often end without any more repetition. When this condition fails to hold, in other words, if "data[index1]" is null or if "index2" has reached "data[index1].length", blank E is executed, "index1" is incremented, and the repetition that follows the "for" statement is executed (note that here, "index1++" in the "for" statement is executed before the program evaluates the condition formula "index1 < data.length"). This suggests that the "for" statement here is for the purpose of moving to the next array when the array in question ("data[index1]") is null or when its end is reached. In order to go on to the next array, "index1" needs to be incremented and "index2" needs to be returned to 0. Since there is not yet this process of returning "index2" to 0, this is what needs to be inserted in blank E. Therefore, the correct answer for blank E is (d) "index2 = 0".

Now, when the condition "index1 < data.length" of the "for" statement no longer holds, there is no more array, so by the final "return false;" the "hasNext" method returns "false".

Blank F

This question is about the "next" method in the class "Char2DArrayCharIterator". Like the "next" method in the class "StringCharIterator", this method returns characters stored in the 2-dimensional array "data" one character at a time and must increment the index of the array appropriately in preparation for the next call. The variable to be incremented is "index2", so the correct answer is (f) "data[index1][index2++]".

In this method, "index2" is incremented, and it may appear that when "index2" goes beyond the end of the array, no process is carried out. However, as we saw in our explanation for blank E, when "index2" reaches the end of an array, a process in the "hasNext" method takes over. In the "next" method, the "hasNext" method is executed first before returning a character, so when "index2" has reached the end of an array, the character is returned after "index2" is returned to 0 and "index1" is incremented.

Question 6 Java Program

Q6. Read the following description of a Java program and the program itself, and then answer Subquestions 1 and 2.

[Program Description]

This program is a simulator that simulates the call center operation of Company A. Company A uses this simulator to estimate the time required for a user to connect to an operator when making a call to the call center. A call made to the call center is always responded by a single operator .

Assume for this simulator that as long as an operator is available, no caller has to wait. For instance, assume six calls were made to the call center at 60-second intervals when two operators were available. The duration of each call was 130 seconds, 100 seconds, 150 seconds, 90 seconds, 110 seconds, and 140 seconds respectively. This means that the wait time of the third user was 10 seconds, the fifth user 30 seconds, and the others 0 seconds. (See Figure 1.)
Note that this simulator simulates one second in the real world in 0.1 second.

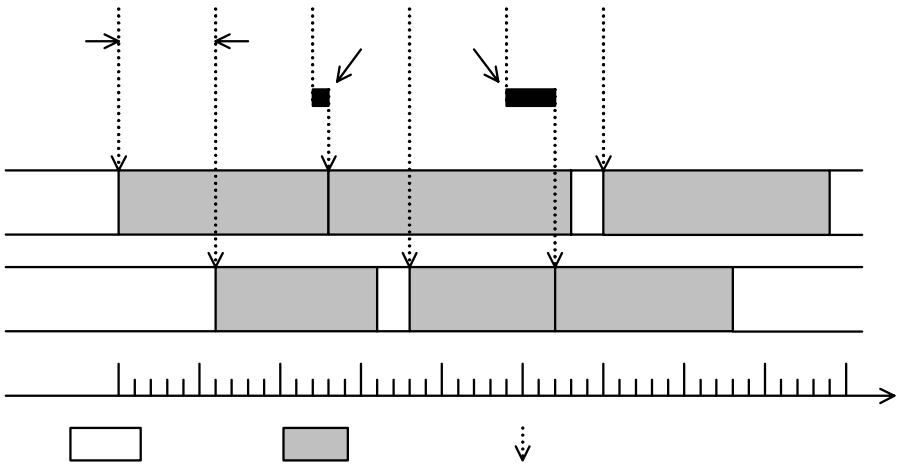


Fig. 1 Users' wait time

This program consists of the following classes:

(1) `CallCenter`

This class represents the call center. It performs simulation after specifying the number of operators , array of talk times, and intervals that generate calls in the arguments of the constructor. It consists of the methods and internal classes described below.

`public static void main(String[] args)`

This method starts the simulator for testing.

`Call answer()`

This method is called by an operator to respond to a call. It returns a `Call` object to the operator. This method makes the operator thread wait until a call can be allocated to the operator. If there remains no call to be allocated after all calls generated by this class have been allocated, it returns `null`.

`Operator`

This thread class simulates the operator. Each operator is processed in an individual thread. It responds to a call to the call center and converses. It repeats the processing until all the calls generated by `CallCenter` have been allocated.

(2) `Call`

This class represents a call from a caller. The duration of talk time is specified in the argument of the constructor. This class consists of the methods described below.

`public void talk()`

This method simulates the status during the talk between an operator and the caller. This method displays the wait time of a caller allocated to the operator in second units (rounded off), and then stops the operator thread only for the time specified as talk time.

Figure 2 shows the results of the program when the example of Figure 1 was simulated.
Note that (s) represents seconds.

```
0(s)
0(s)
10(s)
0(s)
30(s)
0(s)
```

Fig. 2 Results of simulation

`java.util.Vector` in the program is a class that represents a variable-length array, and this class consists of the following methods:

`public boolean add(Object obj)`

It appends element `obj` at the end of the array.

`public Object remove(int index)`

It deletes the element at the position specified by `index` from the array, and returns the deleted element. The position of the first element is 0. The elements after `index` are shifted to the front of the array.

`public boolean isEmpty()`

If there is no element, this method returns `true`. Otherwise, it returns `false`.

[Program 1]

```
import java.util.Vector;

public class CallCenter {
    private final Vector waitingList = new Vector();
    private boolean running; // true when call generated

    public static void main(String[] args) {
        int op = 2;       // Number of operators
        // Talk time (seconds)
        long[] duration = {130, 100, 150, 90, 110, 140};
        long interval = 60;  // Interval for generating call (seconds)
        new CallCenter(op, duration, interval);
    }

    public CallCenter(int op, long[] duration, long interval) {
        running = true;
        // Generate an operator thread and start it.
        for (int i = 0; i < op; i++) new Operator().start();
        long nextCallTime = System.currentTimeMillis();
        for (int i = 0; i < duration.length; i++) {
            // Generate a call and add it to list.
            synchronized (waitingList) {
                waitingList.add(new Call(duration[i]));
                waitingList.notify();
            }
            // Wait until next call is generated.
            nextCallTime += interval * 100;  // Operates 10 times faster.
            long sleeping = [ A ];
            try {
                if (sleeping > 0) Thread.sleep(sleeping);
            } catch (InterruptedException ie) {}
        }
        // End all operator threads.
        running = false;
        [ B ] {
            waitingList.notifyAll();
        }
    }
```

```
    Call answer() {
        synchronized (waitingList) {
            while (waitingList.isEmpty() [ C ]) {
                try {
                    [ D ];
                } catch (InterruptedException ie) {}
            }
            if (waitingList.isEmpty()) return null;
            return (Call)waitingList.remove(0);
        }
    }

    class Operator  extends Thread {
        public void run() {
            Call call;
            [ E ] call.talk();
        }
    }
}
```

[Program 2]

```
public class Call {
    private final long start, duration;
    public Call(long duration) {
        this.duration = duration;
        start = System.currentTimeMillis();
    }
    public void talk() {
        long elapsed = System.currentTimeMillis() - start;
        // Display elapsed time after rounding it off.
        System.out.println([ F ] + "(s)");
        try {
            Thread. sleep(duration * 100);      // Operates 10 times faster.
        } catch (InterruptedException ie) {}
    }
}
```

Subquestion 1

From the answer groups below, select the correct answers to be inserted in the blanks ☐ in the above programs.

Answer group for A:

a) `nextCallTime`
b) `nextCallTime - duration[i] * 100`
c) `nextCallTime - System.currentTimeMillis()`
d) `System.currentTimeMillis() - nextCallTime`

Answer group for B:

a) `for (int i = 0; i < op; i++)`
b) `if (waitingList != null)`
c) `synchronized (this)`
d) `synchronized (waitingList)`

Answer group for C:

a) `&& !running`
b) `&& running`
c) `== !running`
d) `|| running`

Answer group for D:

a) `notify()`
b) `wait()`
c) `waitingList.notify()`
d) `waitingList.wait()`

Answer group for E:

a) `if ((call = answer()) != null)`
b) `if (answer() != null)`
c) `while ((call = answer()) != null)`
d) `while (answer() != null)`

Answer group for F:

a) `(elapsed + 50) / 100`
b) `(elapsed + 500) / 100`
c) `(elapsed - 50) / 100`
d) `(elapsed - 500) / 100`

Subquestion 2

A `CallCenter` instance was generated, as shown below, from the answer group below, select the number of operators occupied after 80 seconds (8 seconds in real time) have elapsed in the simulator.

```
new CallCenter(3, new long[]{70, 90, 100, 110}, 30);
```

Answer group:

a) 0 b) 1 c) 2 d) 3

Answer 6

Call Center Simulator

Correct Answer

[Subquestion 1] A – c, B – d, C – b, D – d, E – c, F – a

[Subquestion 2] c

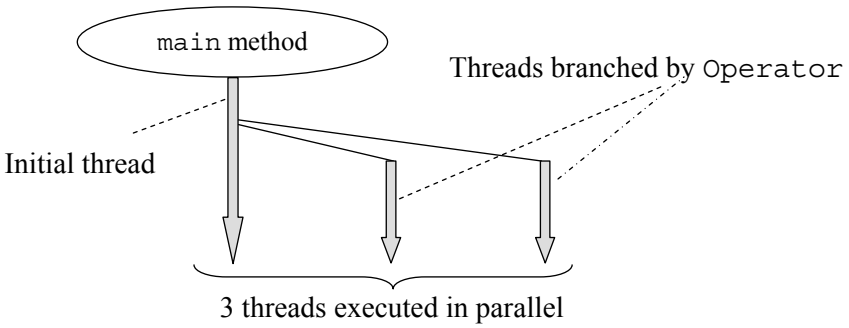
Comments

This is a simulation program for estimating the waiting time from when a phone call is made to a call center to when the caller is connected to an operator . This was the first time a program using Java's multi-thread function appeared on the exam. The question not only asks about the grammatical aspect of Java's thread function, but also requires in depth knowledge of the harmonious operation between threads. Grammatically, methods like "`wait`" and "`notify`", as well as internal classes, are used. Overall, the difficulty level of this question seems to be rather high.

[Program Description]

We will first organize an overview of [Program Description] and the program contents. The program consists of the `CallCenter` class, the `Operator` class (which is an internal class of the `CallCenter` class), and the `Call` class. Because the `Operator` class inherits the `Thread` class, new threads can be started by generating instances and executing the "`start`" method.

When the program is executed from the "`main`" method in the "`CallCenter`" class, the "`main`" method generates "`CallCenter`" objects. These objects are not necessarily stored in special variables to be used, but through the generation of objects, the constructor for the "`CallCenter`" class is called. This constructor for the "`CallCenter`" class first generates instances in the "`Operator`" class and calls the "`start`" method. This begins a new thread. The number of new threads to be generated is determined by the argument "`op`" of the constructor. Here, since `op = 2`, together with the initial thread, there will be a total of three threads to be executed in parallel.



A thread generated by an `Operator` object calls its own "`run`" method. When looking at "`run`" methods of the `Operator` class, we only see blank E and the calling of the "`talk`" method of the "`Call`" object named "`call.talk`". However, from the answer group for E, we see that blank

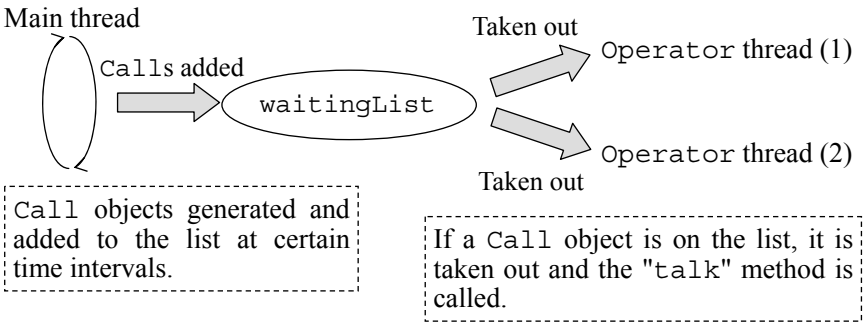
E is calling the "answer" method of the class "CallCenter". The " Operator" class is an internal class of the "CallCenter" class, so the " Operator" objects can reference and use private instance variables and instance methods of CallCenter objects.

Looking now at the contents of the "answer" method of the CallCenter class, we note the following: It checks whether "waitingList" is empty by using a "while" statement. At the end of the method, the statement "return (Call)waitingList.remove(0);" is used to remove the first "Call" object stored in "waitingList" from the list; at the same time, that value is returned as the return value. This portion contains blanks C and D, so the details of the method are difficult to understand, but at least you can see that, if the list contains a "Call" object, it is returned.

Here, once again we need to keep in mind that all processes after the "run" method in the "Operator" class are executed after another thread which is separate from the main thread (thread started from the "main" method) is generated.

The main thread is being processed continuously after the "Operator" objects are generated and the "start" method is called, i.e., even while the "run" method is called and is executing its process. Tracing the processes following the constructor of the "CallCenter" class, we see that "Call" objects are generated in the repetition by a "for" statement and are added to "waitingList". Toward the end of the repetition, the execution is suspended by the "Thread.sleep" method. Based on the text and the comment "Wait until next call is generated" found in the program, calling of this "Thread.sleep" method appears to be for generating calls ("Call" objects) at certain time intervals.

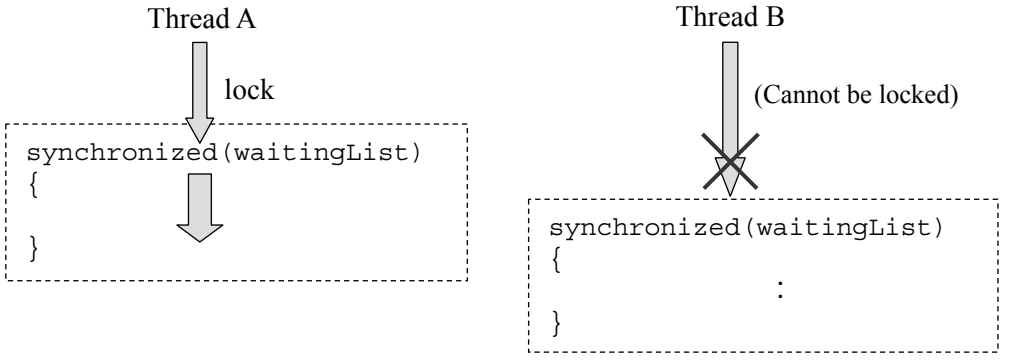
From these contents, we understand that, in this program the main thread adds "Call" objects to "waitingList" at certain time intervals while threads by "Operator" take these "Call" objects from "waitingList" by the "answer" method and call the "talk" method.



Once you understand the operation of this program thus far, then the only thing you have to consider is how to control possible conflicts between threads. In the program, each thread is referencing "waitingList", so you need to consider the problem of conflicting among the threads regarding this "waitingList". For instance, suppose a thread has checked that the list is not empty and is about to take out an element. But just in that little time period between checking that it is not empty and taking out an element, another thread can remove the element, causing the list to be empty. This leads to an error. In multi-thread programming, a problem can arise when multiple threads attempt to gain access to the same information.

To handle this problem, in Java, you can practice synchronization control between threads by using a "synchronized statement". A synchronized statement locks a designated object, which remains in effect from the time when the thread enters the synchronized block to the time when it leaves the block. While one thread has an object locked, another thread cannot lock the

same object by synchronized statement; that thread goes into standby until the first thread releases its lock.



While one thread has "waitingList" locked with a "synchronized" statement, another thread cannot enter a "synchronized" statement.

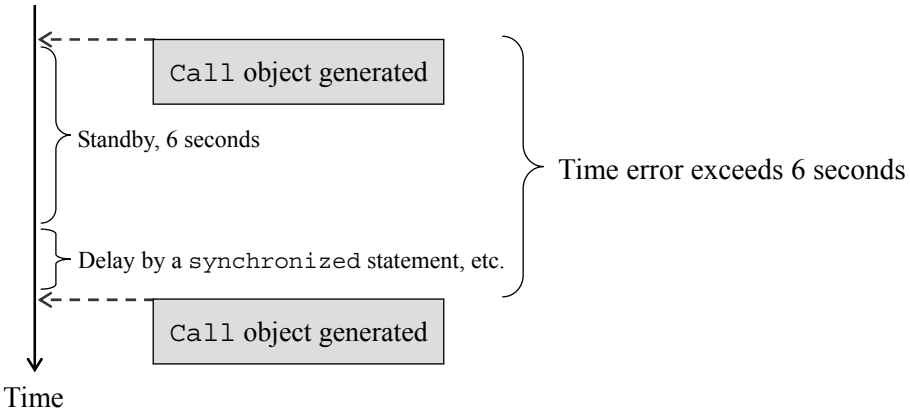
For example, as with the "answer" method in the "CallCenter" class, synchronization control must be performed using a "synchronized" statement in cases where the status of "waitingList" is confirmed by the "isEmpty" method and is then updated by a method such as "remove". .

[Subquestion 1]

Blank A:

You are asked to provide the value to which the variable "sleeping" (long-type) should be set. Since this value is designated as the argument for "thread.sleep", it is the sleeping time for the thread. As explained in [Program Description], the "for" statement seen here is probably a loop for the main thread to generate calls ("Call" objects) at certain time intervals. For instance, calls at 60-second time intervals can be simulated if we generate Call objects every 6 seconds by simulator (an actual time period of 1 second is simulated by 0.1 second on the simulator). The argument of the "sleep" method is defined in milliseconds, so the argument is 6000 for 6-second intervals. "interval * 100" found in the program is for this argument. Here, the variable "interval" is an argument given to the constructor and has the value 60. 100 means "100 milliseconds" (normally, 1 second = 1000 milliseconds, but since 1 second is simulated by 0.1 second [= 100 milliseconds], the multiplier is 100). However, this expression "interval * 100" is used in the statement "nextCallTime += interval * 100" as the quantity to be added to the term "nextCallTime". This cannot be found among the options in the answer group.

How do we interpret this? This is a problem, but consider that there may be a waiting time caused by the "synchronized" statement in the beginning of the "for" statement and that it takes time to generate Call objects and add them to "waitingList". Now it makes more sense. Suppose that 6000 (= 6 seconds) is entered as the argument of the "sleep" method. If the "synchronized" statement triggers synchronization control, thus producing waiting time, it will take a longer time than a 6-second interval from the generation of one "Call" object to the generation of the next "Call" object. If this process is repeated multiple times, the error will accumulate, eventually resulting in erroneous generation times for Call objects.



To avoid this, the program introduces a variable called "nextCallTime", which indicates the time when a subsequent "Call" object is to be generated. When the first "Call" object is generated, "nextCallTime" is initialized to the current time at that point (the method "System.currentTimeMillis" obtains the current time to milliseconds), and the times for a subsequent object generation are calculated at 6-second intervals after that current time. Since generating "Call" objects may be taking some time also, the sleeping time by the "sleep" method should be the difference between the moment when an object is to be generated next and the current time. The correct answer for blank A is, therefore, (c), "nextCallTime – System.currentMillis()".

Blank B:

Inside the block where blank B is located, we read "waitingList.notifyAll();". The method "notifyAll" is one of the methods defined in the "Object" class. It notifies all threads placed on standby by the "wait" method to resume the threads. Like "synchronized", "wait" and "notifyAll" are used when threads are to be synchronized. "notifyAll" (or "notify") notifies all threads waiting in the "wait" method of a certain condition for which they are to resume their operation; this is how "notifyAll" performs synchronization. Like a "synchronized" statement, these methods are used to lock and unlock certain objects. In other words, they call "wait" for "waitingList" objects and, to notify those threads placed on standby, again they call the "notifyAll" method for "waitingList" objects. Further, these methods must be used while "waitingList" is locked by a "synchronized" statement . Otherwise, an exception "IllegalMonitorStateException" occurs. Hence, a "synchronized" statement for "waitingList" must be inserted in blank B, and the answer is (d).

Blanks C and D:

These blanks regard the "answer" method. As explained in [Program Description], this method is called by the "Operator" thread and, if a Call object exists in "waitingList", then it returns the object. Blank C is a statement added to the condition for this "while" statement. Blank D is the process to be performed while this condition is satisfied. Consider D first. This is what needs to be executed repeatedly while "waitingList" is empty, so it is expected to be the spot where threads are placed on standby by the "wait" method. This explains the following portion in the constructor of the class "CallCenter" as well.

```
              :
synchronized(waitingList) {
   waitingList.add(new Call(duration[i]));
   waitingList.notify();
}
              :
```

In other words, the "Operator" thread calls the "answer" method to check whether "waitingList" is empty and, if it is, waits until the main thread adds a "Call" object to "waitingList". Meanwhile, after adding an object to "waitingList", the main thread calls "notify" to notify that an object has been added and to resume threads that had been on standby. The "notify" method is different from "notifyAll" in that only one thread is notified even when there are multiple threads. The "notify" method is called for "waitingList" objects, so the "wait" method needs to be called for "waitingList" as well. Hence, the correct answer for blank D is (d).

Let us now go back to blank C. From the options in the answer group, you know that this condition involves a variable called "running". The variable "running" is first initialized to "true" at the beginning of the "CallCenter" constructor and is changed to "false" after the "for" loop ends, along with the comment "End all Operator threads". Here, the "notifyAll" method is also called. Hence, the meaning of the variable "running" seems to imply that while "running" is "true", the main thread is operating, and "Call" objects are generated and added to "waitingList". When the main thread terminates, the variable "running" is turned to "false". Consider the multiple threads that have been placed on standby by the "wait" method. All processes seem to make sense if we use "notifyAll" instead of "notify" so that all of these threads are notified. While the main thread is running, the variable "running" is "true", so blank C should be (b) "&& running". Then, when "running" is changed to "false" by the main method, this "while" loop exits. If, at that time, "waitingList" is empty, the "answer" method returns "null" as its return value, so the whole operation satisfies the description of the "answer" method of the question.

Blank E:

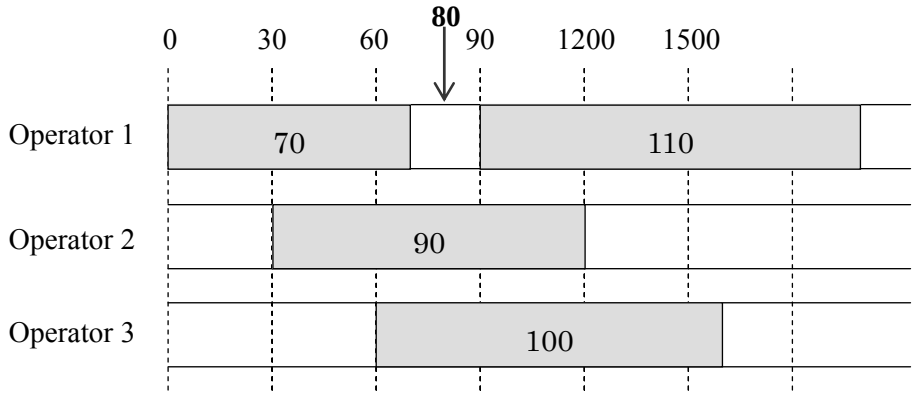
This is in the "run" method in the "Operator" class. Here, the "answer" method is called, and its return value must be stored in the variable "call". Otherwise, an exception occurs in the immediately following statement "call.talk();". Therefore, the choices can be narrowed down to (a) and (c). If (a) is picked, after executing it once, the "run" method ends, and the thread also ends at the same time. However, considering the nature of the question, this thread must continue to receive "Call" objects repeatedly while the "answer" keeps returning "Call" objects. Hence, the correct answer is (c). If the "answer" method returns "null", it suggests that the main thread has stopped generating "Call" objects; then, the "Operator" thread also ends.

Blank F:

What is to be displayed here is the waiting time of the user assigned to an ooperator . The line preceding it is obtaining the result of subtracting "`start`" from the current time as the variable "`elapsed`". On the other hand, the variable "`start`" is set by the object constructor to the current time, i.e., time when the object was generated. Consider this "`start`" time as the time when the user made the phone call, and consider the time when this "`talk`" method was executed as the time when the operator answered that call; the difference is the user's waiting time. Stated differently, the variable "`elapsed`" retains the user's waiting time. This variable "`elapsed`" stores time in milliseconds. In this simulation, 0.1 second (= 100 milliseconds) is considered 1 second of the actual world, so the number of elapsed seconds in the real world can be calculated by dividing the time by 100. However, the comment in the immediately preceding line says "after rounding it off" (rounding to the nearest second). So we need to select a formula that gives us the correctly rounded value. In Java, when an integer is divided by another integer, the fractional (decimal) part gets truncated, so adding 50 to a number in advance produces the same result as rounding to the nearest whole number when divided by 100. Hence, the answer is (a) "`(elapsed + 50) / 100`".

[Subquestion 2]

For arguments of the constructor in the "`CallCenter`" class, we designate the number of operators , array of call durations, and the length of time intervals after which calls are made. Plugging the arguments given in the question into this format, we note that there are 3 operators , that calls are made every 30 seconds, and that the calls last 70, 90, 100, and 110 seconds, respectively. The following figure shows the simulation result under these conditions.



The above figure indicates that the number of operators on the phone after 8 seconds have elapsed in the simulator (80 seconds in real time) later is 2, so the correct answer is (c).

FE Exam Preparation Book Vol.2

Preparation for Afternoon Exam

First Edition: January, 2008

This book, prepared by Information-Technology Promotion Agency, Japan (IPA) with approval, is a modified, English translation of the books published by ITEC Co., LTD.

2006 基本情報技術者　午後問題の重点対策（ISBN: 978-4872685022）
2005 春　基本情報技術者 徹底解説本試験問題（ISBN: 978-4872684001）