



October 2016

Fundamental IT Engineer Examination (Afternoon)

Questions must be answered in accordance with the following:

Question Nos.	Q1 – Q6	Q7 , Q8
Question Selection	Compulsory	Select 1 of 2
Examination Time	13:30 – 16:00 (150 minutes)	

Instructions:

1. Use a pencil. If you need to change an answer, erase your previous answer completely and neatly. Wipe away any eraser debris.
2. Mark your examinee information and test answers in accordance with the instructions below. Your answer will not be graded if you do not mark properly. Do not mark or write on the answer sheet outside of the prescribed places.

(1) **Examinee Number**

Write your examinee number in the space provided, and mark the appropriate space below each digit.

(2) **Date of Birth**

Write your date of birth (in numbers) exactly as it is printed on your examination admission card, and mark the appropriate space below each digit.

(3) **Question Selection**

For **Q7** and **Q8**, mark the **(S)** of the question you select to answer in the “Selection Column” on your answer sheet.

(4) **Answers**

Mark your answers as shown in the following sample question.

[Sample Question]

In which month is the spring Fundamental IT Engineer Examination conducted?

Answer group

- a) September b) October c) November d) December

Since the correct answer is “b) October”, mark your answer sheet as follows:

[Sample Answer]

Sample	<input type="radio"/> a	<input checked="" type="radio"/> b	<input type="radio"/> c	<input type="radio"/> d	<input type="radio"/> e	<input type="radio"/> f	<input type="radio"/> g	<input type="radio"/> h	<input type="radio"/> i	<input type="radio"/> j
--------	-------------------------	------------------------------------	-------------------------	-------------------------	-------------------------	-------------------------	-------------------------	-------------------------	-------------------------	-------------------------






Do not open the exam booklet until instructed to do so.

Inquiries about the exam questions will not be answered.

Notations used for pseudo-language

In questions that use pseudo-language, the following notations are used unless otherwise stated:

[Declaration, comment, and process]

Notation		Description
○		Declares names, types, etc., of procedures, variables, etc.
/* text */		Describes comments in the text.
Process	<ul style="list-style-type: none"> variable ← expression 	Assigns the value of the expression to the variable.
	<ul style="list-style-type: none"> procedure(argument, ...) 	Calls the procedure and passes / receives the argument.
		Indicates a one-way selection process. If the conditional expression is true, then the process is executed.
		Indicates a two-way selection process. If the conditional expression is true, then process 1 is executed. If it is false, then process 2 is executed.
		Indicates a pre-test iteration process. While the conditional expression is true, the process is executed repeatedly.
		Indicates a post-test iteration process. The process is executed, and then while the conditional expression is true, the process is executed repeatedly.
		Indicates an iteration process. The initial value init (given by an expression) is stored in the variable at the start of the iteration process, and then while the conditional expression cond is true, the process is executed repeatedly. The increment incr (given by an expression) is added to the variable in each iteration.

[Logical constants]

true, false

(continued on next page)

[Operators and their priorities]

Type of operation	Operator	Priority
Unary operation	+, -, not	<div style="text-align: center;"> High ↑ ↓ Low </div>
Multiplication, division	×, ÷, %	
Addition, subtraction	+, -	
Relational operation	>, <, ≥, ≤, =, ≠	
Logical product	and	
Logical sum	or	

Note: With division of integers, an integer quotient is returned as a result.

The “%” operator indicates a remainder operation.

Questions **Q1** through **Q6** are all **compulsory**. Answer every question.

Q1. Read the following description of security vulnerability, and then answer Subquestions 1 and 2.

TLS (Transport Layer Security) and SSL (Secure Sockets Layer) are cryptographic protocols that provide communications security between a client and server over the Internet. Although TLS is the successor of SSL, they are usually referred to together as SSL/TLS.

[The handshake protocol]

Figure 1 shows the flow of information and processes between the client and server before the secure SSL/TLS session is established.

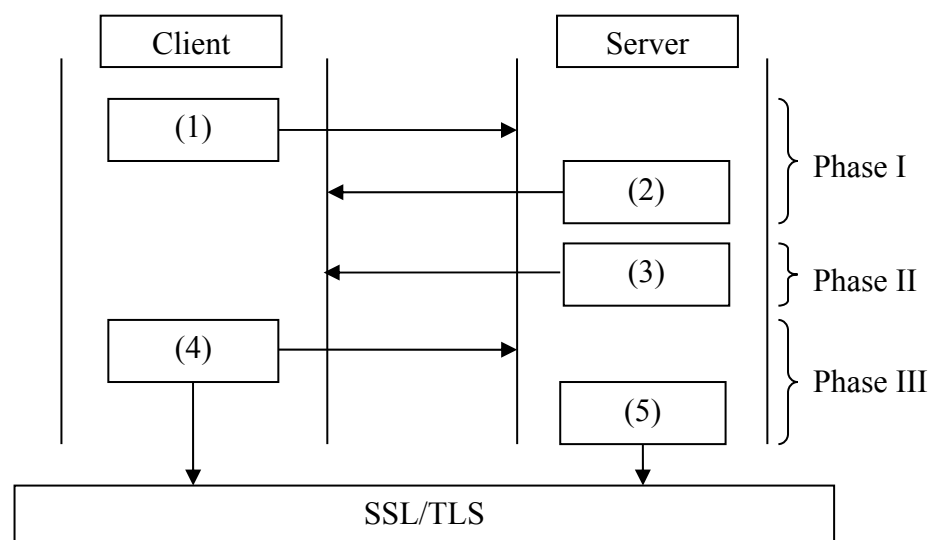


Figure 1 Overview of SSL/TLS handshake protocol

- (1) In Phase I, the client sends the Client Hello message that contains the highest SSL version supported by the client, a random number to be used for master secret generation, and the list of encryption and compression algorithms supported by the client.
- (2) Next, the server responds with a Server Hello message that contains the highest SSL version supported by the client and server, another random number for master secret generation, and the selected encryption and compression algorithm to be used.
- (3) In Phase II, in most cases, the server sends its certificate (or certificate chain) to the client. At this stage, the **A** is available to the client. If required, the server can send a certificate request to the client as well.

- (4) In Phase III, if explicitly requested, the client sends its certificate and authenticates itself with the server. Then, depending on the key exchange algorithm, the client generates its contribution to the pre-master secret. The client encrypts it with the **A** from the server's certificate, and sends it to the server. Now, the client has all the information required to generate the master secret key using the value generated on the client side and the information obtained in previous steps.
- (5) Next, the server uses the **B** to decrypt the value received from the client. It can now also generate the master secret key. Thus the handshake on the server side is complete.

[Heartbleed Bug]

Heartbleed is vulnerability in OpenSSL, an open source implementation of the SSL/TLS protocol. The name heartbleed is derived from the heartbeat extension used to maintain the SSL/TLS session alive. Because it is costly to initiate a new SSL/TLS session, maintaining alive current sessions between the server and client for a reasonable period is a viable solution especially in Web sites and servers with a high volume of SSL/TLS traffic.

The heartbeat protocol uses two message types: Heartbeat Request and Heartbeat Response. Either side can send a heartbeat request that contains a payload message and the size of the message itself, and the other side will respond with the same payload message to indicate that it is alive. Figure 2 shows the heartbeat requests and responses, with "Hello" as the payload message, of a normal client N, a malicious client M, and a possible victim client V.

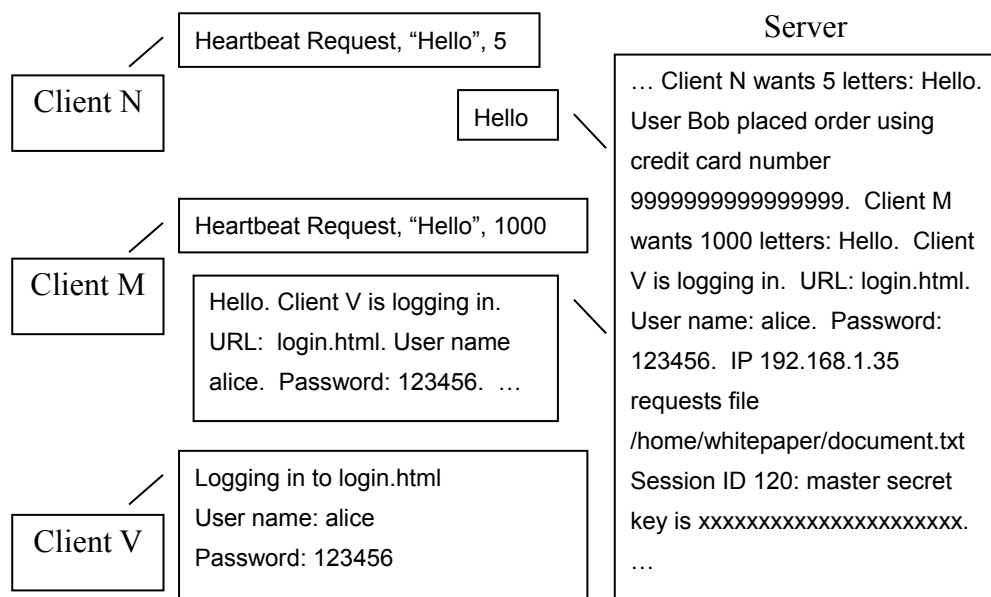


Figure 2 Example of heartbleed requests and responses

The heartbleed bug exploits the heartbeat response message by incorrectly specifying the size of the payload. For client M, the server failed to perform bound checking and returns 1000 bytes of its memory starting from the location of the payload instead of simply sending the payload itself. Because the payload size is 16-bit binary value, attackers can obtain up to of the server memory with each heartbeat request. Therefore, it is possible for the server to leak sensitive information. In the worst-case scenario, if the is leaked, attackers can easily decrypt all captured communications, both past and future. It also allows the attackers to impersonate as the server even if the bug is already patched.

The bug was discovered in 2014 in the older version of OpenSSL, and the new version 1.0.1g, which fixed the bug was released in April 2014. However, the bug was widespread because OpenSSL is integrated with various versions of operating systems that range from servers to mobile devices including popular Web server software. Therefore, users are urged to upgrade to the latest version of OpenSSL.

Subquestion 1

From the answer groups below, select the correct answer to be inserted in each blank in the above description. If needed, select the same answer twice or more.

Answer group for A, B and D

- | | |
|-------------------------|------------------------|
| a) client's private key | b) client's public key |
| c) pre-master secret | d) master secret key |
| e) server's private key | f) server's public key |

Answer group for C

- | | | |
|---------------|--------------|--------------|
| a) 256 bytes | b) 32k bytes | c) 64k bytes |
| d) 256k bytes | e) 1M bytes | f) 4M bytes |

Subquestion 2

From the answer group below, select two appropriate statements that concern the heartbleed bug.

Answer group

- a) Both a client and server can send heartbeat requests. Thus, it is possible for a malicious user on the server side to read data from the client's memory as well.
- b) It is a flaw in the design of the heartbeat extension. In order to avoid losing sensitive data, a user should avoid sending heartbeat requests to servers until further notice.
- c) It is difficult to investigate an attack because an attacker only reads the contents of the memory and the attack leaves no trail of damage.
- d) On vulnerable servers, an attacker can send illegal heartbeat request messages to alter the contents of the memory and gain root access to the operating system.
- e) The bug was discovered in 2014 in the older version of OpenSSL. Therefore, it can be concluded that there were no attacks prior to the discovery.

Q2. Read the following description of arithmetic circuit for multiplication, and then answer Subquestions 1 and 2.

Hardware engineers are planning to design an arithmetic circuit that multiplies two fixed-point binary numbers in signed magnitude representation.

[Signed magnitude representation]

In signed magnitude representation, a fixed-point binary number is represented as follows:

Sign bit	Magnitude bits
----------	----------------

The leftmost bit is the sign bit: 0 means positive, 1 means negative. The rest of the bits after the sign bit are the magnitude bits that represent the absolute value of the number.

Examples:

01001 represents +9 because the sign bit is 0 (+) and the magnitude bits are 1001 (9).

10011 represents -3 because the sign bit is 1 (-) and the magnitude bits are 0011 (3).

In signed magnitude representation, there are two zero representations: +0 and -0. When the magnitude bits are all 0s, the value is evaluated as 0 regardless of the sign bit value.

[Arithmetic circuit for multiplication]

- (1) Calculation is performed in signed magnitude representation.
- (2) A multiplicand and multiplier are 5 bits in length (including the sign bit), and the product is 9 bits in length (including the sign bit).
- (3) There are 6 registers: M_S (1-bit register), Q_S (1-bit register), P_S (1-bit register), M_M (8-bit register), Q_M (4-bit register), and P_M (8-bit register).

The following steps describe how the multiplication operation proceeds, with $(-3) \times (+9) = (-27)$, whose internal representation is $10011 \times 01001 = \boxed{A}$, as an example.

- [1] Load the sign bit of the multiplicand to M_S .
- [2] Load the magnitude bits of the multiplicand to bit positions 1 through 4 of M_M . Set 0s to the rest of the bits of M_M (shaded parts in Figure 1).
- [3] Load the sign bit of the multiplier to Q_S .
- [4] Load the magnitude bits of the multiplier to Q_M .

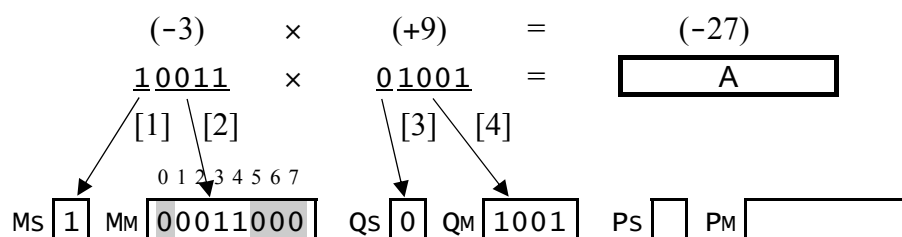


Figure 1 Execution results of steps [1] through [4]

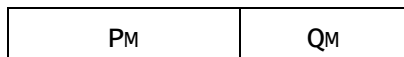
- [5] Test the rightmost bit of Q_M . If it is 1, copy the contents of M_M to P_M . If it is 0, load 00000000 to P_M .
- [6] Repeat [6-1] through [6-3] three times.
- [6-1] Shift the contents of P_M 1-bit to the right.
- [6-2] Shift the contents of Q_M 1-bit to the right.
- [6-3] Test the rightmost bit of Q_M . If it is 1, add the contents of M_M to P_M .
- [7] Set the result of the logical operation B to P_S .
- [8] Finally, the sign bit of the product is obtained in P_S , and the magnitude bits of the product are obtained in P_M .

After the execution of [5]:	Q_M	1001	P_M	00011000
After the first iteration of [6]:	Q_M	0100	P_M	
After the second iteration of [6]:	Q_M	0010	P_M	C
After the third iteration of [6]:	Q_M	0001	P_M	

Note: Shaded parts are not shown.

Figure 2 Execution results of steps [5] and [6]

In real implementation, if P_M and Q_M are combined into one longer register as shown below, steps [1] through [8] still work correctly, and the number of shift operations can be halved.



Subquestion 1

From the answer groups below, select the correct answer to be inserted in each blank in the above description. Here, the symbols “&”, “|”, and “^” denote the logical operators AND, OR, and XOR, respectively.

Answer group for A

- | | |
|--------------|--------------|
| a) 000011011 | b) 000110011 |
| c) 100011011 | d) 100110011 |

Answer group for B

- | | | |
|---------------------|--------------------|-------------------------|
| a) ($M_S \& Q_S$) | b) ($M_S Q_S$) | c) ($M_S \wedge Q_S$) |
|---------------------|--------------------|-------------------------|

Answer group for C

- | | |
|-------------|-------------|
| a) 00000011 | b) 00000110 |
| c) 00001100 | d) 01100000 |

Subquestion 2

From the answer group below, select the correct answer to be inserted in each blank in Figure 3.

In the multiplication operation, step [6] is executed 3 times. The hardware engineers work develop the design for a logic circuit for loop control as shown in Figure 3.

In Figure 3, the 4-bit counter (hereinafter, the counter) functions as follows:

- First, with the load signal, counter value 3 is set in the counter.
- Then, each time the counter receives the decrement signal, the counter decrements the counter value by 1, and outputs the decremented value on lines O8, O4, O2 and O1.
 - 1st time, $(O8, O4, O2, O1) = (0, 0, 1, 0)$ because the decremented value is 2 (=0010).
 - 2nd time, $(O8, O4, O2, O1) = (0, 0, 0, 1)$ because the decremented value is 1 (=0001).
 - 3rd time, $(O8, O4, O2, O1) = (0, 0, 0, 0)$ because the decremented value is 0 (=0000).

The logic circuit in Figure 3 finally outputs the signal Z, which is a binary value (0 or 1). The logic circuit outputs $Z = 1$ when $(O8, O4, O2, O1) = (0, 0, 0, 0)$. Otherwise, it outputs $Z = 0$.

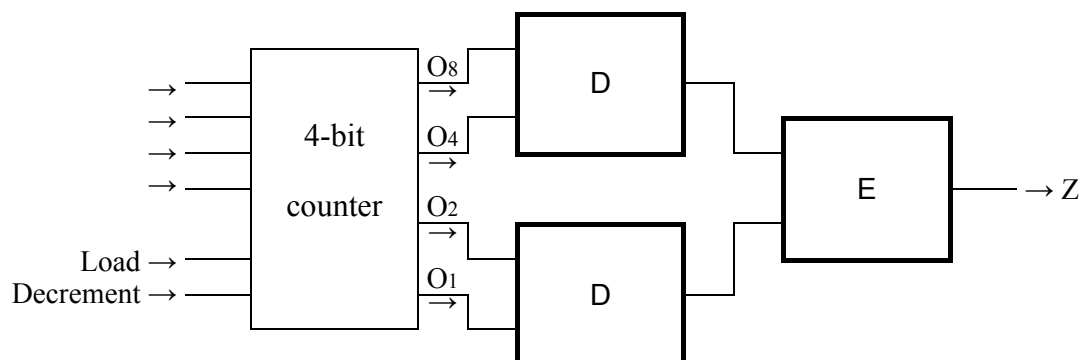


Figure 3 Logic circuit for loop control

Answer group for D and E

- a) (AND gate) b) (OR gate) c) (XOR gate)
- d) (NAND gate) e) (NOR gate)

Q3. Read the following description of a database system, and then answer Subquestions 1 through 3.

The Lucky Dog Grooming Parlor is a pet care shop that provides full pet-styling salon services with several facilities, especially for dogs. The shop maintains information about each pet in a table named PetTable with attributes that include each dog's ID, name, breed, pet owner's name, and the balance due on services. The table structure is as follows:

PetTable (DogID, DogName, Breed, OwnerName, BalanceDue)

Subquestion 1

From the answer group below, select the correct answer to be inserted in the blank in the following SQL statement.

Some pet owners own more than one dog. The shop manager wants to generate a report that displays a list of pet owners who own more than one dog. For this purpose, the SQL statement "SQL1" is created.

```
-- SQL statement "SQL1"
SELECT  OwnerName, DogName, Breed, BalanceDue
FROM    PetTable
WHERE    A
ORDER BY OwnerName
```

An example of the report created by "SQL1" is as follows:

OwnerName	DogName	Breed	BalanceDue
Henry Chauncey	Buddy	Great Dane	1000
Henry Chauncey	Abe	Bulldog	300
Mike Barz	Baxter	Boxer	1000
Mike Barz	Fluffy	Poodle	0
Mike Barz	Love	Poodle	100

Answer group for A

- a) COUNT(OwnerName)>1
- b) DogID IN (SELECT DogID FROM PetTable HAVING(COUNT(DogID)>1))
- c) DogID IN (SELECT DogID FROM PetTable HAVING(COUNT(OwnerName)>1))
- d) HAVING(COUNT(DogID)>1)
- e) OwnerName IN (SELECT OwnerName FROM PetTable
GROUP BY OwnerName HAVING(COUNT(OwnerName)>1))

Subquestion 2

From the answer group below, select the correct answer to be inserted in each blank in the following SQL statement. If needed, select the same answer twice or more.

The Lucky Dog Grooming Parlor also wants to maintain information about each pet owner in a table named OwnerTable with attributes that include each owner's ID, name, address, township, and telephone number. The table structure is as follows:

OwnerTable (OwnerID, OwnerName, Address, Township, TelephoneNo)

To obtain a relationship between OwnerTable and PetTable, the attribute OwnerName in PetTable is replaced by the attribute OwnerID. Consequently, the table structure of PetTable is as follows:

PetTable (DogID, DogName, Breed, OwnerID, BalanceDue)

The shop manager wants to give 10% discount on the current balance due for pet owners who come from any township different from the "Wild Rose" township, which is where the shop is located. For this purpose, the SQL statement "SQL2" is created.

```
-- SQL statement "SQL2"
UPDATE  B
  SET BalanceDue = BalanceDue - (BalanceDue * 0.1)
  WHERE  C IN (SELECT  D
                        FROM  E
                        WHERE Township <> 'Wild Rose')
```

The following table shows how the balance due is updated by "SQL2".

DogID	DogName	Breed	BalanceDue before update	BalanceDue after update	Township
1	Buddy	Great Dane	1000	1000	wild Rose
2	Abe	Bulldog	300	300	wild Rose
3	Acridus	Great Dane	1500	1350	Schaumburg
4	Bam Bam	Bulldog	1000	900	Schaumburg
5	Baxter	Boxer	1000	900	Dubuque
6	Fluffy	Poodle	0	0	Dubuque
7	Love	Poodle	100	90	Dubuque

Answer group for B through E

- a) BalanceDue b) DogID c) OwnerID
- d) OwnerTable e) PetTable f) Township

Subquestion 3

From the answer group below, select the correct answer to be inserted in each blank in the following SQL statement.

The shop manager also wants to generate a report that displays a list of the number of dogs and the total amount of balance dues by each township. For this purpose, the SQL statement “SQL3” is created.

```
-- SQL statement "SQL3"  
SELECT  F  
      FROM  G  
GROUP BY OwnerTable.Township
```

An example of the report created by “SQL3” is as follows:

NoOfDogs	TotalAmount	Township
2	1300	wild Rose
2	2250	Schaumburg
3	990	Dubuque

Answer group for F and G

- a) COUNT(DogID) AS NoOfDogs,
SUM(BalanceDue) AS TotalAmount, Township
- b) COUNT(DogID) AS NoOfDogs,
TOTAL(BalanceDue) AS TotalAmount, Township
- c) OwnerTable
- d) OwnerTable INNER JOIN PetTable
ON OwnerTable.OwnerID = PetTable.OwnerID
- e) PetTable IN (SELECT OwnerID FROM OwnerTable
HAVING PetTable.OwnerID = OwnerTable.OwnerID)

Q4. Read the following description of network connectivity problems, and then answer Subquestions 1 and 2.

Company ABC has established its internal LANs to share resources within the office and have access to the Internet. The office is divided into two sections X and Y with 50 PCs in total. It also has a Web server hosted within the premises of an Internet service provider. For optimizing security and performance, the two sections are separated by a router. Table 1 lists the PCs in sections X and Y. Figure 1 shows the network connectivity map.

Table 1 List of PCs connected via Switch/Hub

PCs	Connecting Switch/Hub	Section
PC01 to PC20	Switch1	X
PC21 to PC30	Hub	X
PC31 to PC50	Switch2	Y

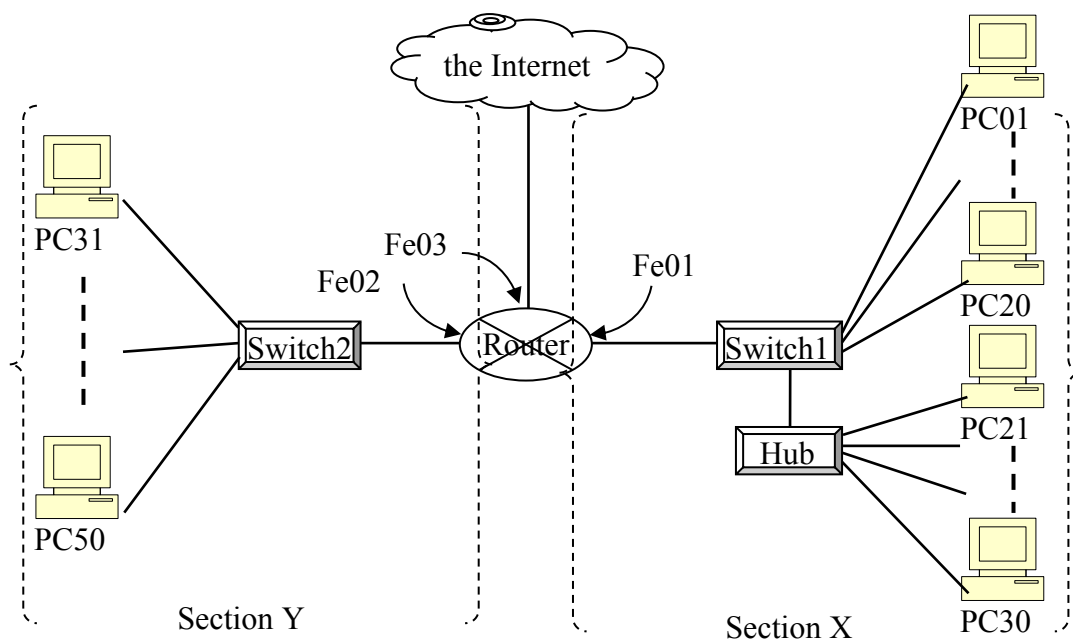


Figure 1 Network connectivity map

The PCs on both LANs are configured by class C private IP addresses with default subnet mask. Table 2 lists partial information on the device name, IP address, subnet mask, MAC address, and the default gateway of the devices.

Table 2 List of devices on internal LANs

Device Name	IP Address	MAC Address	Default gateway	Section
Fe01	192.168.1.1/24	AA:AA:AA:12:34:5A		X
Switch1	192.168.1.11/24	AA:AA:12:5E:7D:6D		
PC01	192.168.1.101/24	AA:0F:3C:33:44:01	192.168.1.1	
...	192.168.1.1	
PC20	192.168.1.120/24	AA:0F:3C:33:44:20	192.168.1.1	
PC21	192.168.1.121/24	AA:0F:3C:33:44:21	192.168.1.11	
...	192.168.1.11	
PC30	192.168.1.130/24	AA:0F:3C:33:44:30	192.168.1.11	Y
Fe02	192.168.3.2/24	AA:AA:AA:12:34:5B		
Switch2	192.168.3.22/24	AA:AA:12:5E:7D:6E		
PC31	192.168.2.231/24	AA:0F:3C:33:44:31	192.168.3.2	
...	192.168.3.2	
PC50	192.168.2.250/24	AA:0F:3C:33:44:50	192.168.3.2	Internet
Fe03	203.129.30.57/30	AA:AA:AA:12:34:5C		
Web server web.example.com	198.51.100.189/28	AB:CD:EF:12:34:56		

Subquestion 1

From the answer group below, select the appropriate answer to be inserted in each blank in the following description.

After the network configuration was completed, the employees of both sections started sharing resources within their own sections. However, most employees currently encounter problems with connectivity among sections and to the Internet. Company ABC hired a network engineer to determine the problems.

The network engineer made a plan to test connectivity using ping commands. He arranged three test cases, and started the testing.

The results of test case (1) are as follows:

Test case	Source	Destination	Command	Result	Cause
(1)	PC01 – PC20	PC21	ping 192.168.1.121	ping successful	
	PC21 – PC30	PC01	ping 192.168.1.101	ping successful	
	PC01	Router Fe01	ping 192.168.1.1	ping successful	
	PC21	Router Fe01	ping 192.168.1.1	ping successful	
	PC01 – PC20	Web Server	ping 198.51.100.189	ping successful	
	PC21 – PC30	Web Server	ping 198.51.100.189	ping failed	<input type="text" value="A"/>

The network engineer resolved cause **A**, and then proceeded with test case (2).

The results of test case (2) are as follows:

Test case	Source	Destination	Command	Result	Cause
(2)	PC01 – PC20	PC31	ping 192.168.2.231	ping failed	B
	PC21 – PC30	PC31	ping 192.168.2.231	ping failed	
	PC31 – PC50	PC01	ping 192.168.1.101	ping failed	
	PC31 – PC50	PC21	ping 192.168.1.121	ping failed	
	PC31 – PC49	PC50	ping 192.168.2.250	ping successful	
	PC31 – PC50	Web Server	ping 198.51.100.189	ping failed	B
	PC31 – PC50	Router Fe02	ping 192.168.3.2	ping failed	

The network engineer resolved cause **B**, and then he proceeded with test case (3).

The results of test case (3) are as follows:

Test case	Source	Destination	Command	Result	Cause
(3)	PC01 – PC50	Web Server	ping web.example.com	ping failed	C

Answer group for A through C

- a) Router port Fe02 was not registered with Fe01 and Fe03.
- b) The DNS server address was not configured in any of the PCs.
- c) The gateway addresses of all the PCs of Company ABC were not assigned to 203.129.30.57.
- d) The gateway addresses of PC21 to PC30 were assigned to 192.168.1.11/24 instead of 192.168.1.1/24.
- e) The IP address of Fe03 was not assigned to 192.168.2.2/24.
- f) The IP address of Switch1 was not assigned to 192.168.1.1/24.
- g) The IP address of Switch1 was not assigned to 192.168.3.2/24.
- h) The IP addresses of PC31 to PC50 were wrongly configured to 192.168.2.xxx instead of 192.168.3.xxx.

Subquestion 2

From the answer group below, select the correct answer to be inserted in the blank in the following description.

An employee working in Section Y has transferred to Section X. He had to move with his PC32 because it contains much information with regard to common company benefits. He connected his PC to Switch1. However, he is neither able to browse the Web sites on the Internet nor use the shared resources in Section X. He noticed that the activity indicator of his PC's Ethernet port, as well as the corresponding switch port, is blinking normally, as other PCs do. He asked the network engineer for a solution to the problem. The network engineer advised him to D. The employee found this helpful.

Answer group for D

- a) change his PC's DNS address to 203.129.30.57 and default gateway to 192.168.1.1
- b) change his PC's IP address to 192.168.1.132 and DNS to 203.129.30.57
- c) change his PC's IP address to 192.168.1.132 and default gateway to 192.168.1.1
- d) replace the straight through UTP cable with a crossover UTP cable
- e) restart Switch1 to reset the MAC address table of the switch

Q5. Read the following description of test design for software, and then answer Subquestions 1 through 3.

Company *N*, a system integrator, is reviewing its testing method in order to reduce the number of bugs left uncorrected in the programs developed by the company.

[Description of the testing method used in Company *N*]

Company *N* mainly tests the programs developed by the company using control flow testing, which is a white box testing method.

Control flow testing focuses on the smallest units that form a program, such as commands, paths, and decision conditions. Test cases and test data are prepared in accordance with the coverage criteria defined during test planning, and the behavior of the developed programs is checked.

The coverage criteria include statement coverage, where all the statements are executed at least once in the test, and decision condition coverage (hereinafter, branch coverage), where all the paths after all the branches are executed at least once.

Company *N* uses branch coverage as its coverage criteria.

[Description of the decision condition of branch coverage used by Company *N*]

A decision condition for a branch includes a single condition that evaluates only one condition, and a multiple condition that evaluates two or more single conditions combined with “and” or “or”. The following example illustrates single conditions and a multiple condition:

Example	$(a > b)$	and	$(a < c)$
	Single		Single
	condition		condition
	Multiple condition		

Here, when a program is executed, short-cut evaluation is applied to a multiple condition. In short-cut evaluation, single conditions that constitute a multiple condition are evaluated in sequence from left to right. Once the result of the multiple condition is determined, the remaining single conditions are not evaluated. For example, in the case of a multiple condition in which two single conditions are combined with “and”, if the evaluation result of the first single condition is false, the evaluation result of the multiple condition is false, regardless of the evaluation result of the second single condition. Therefore, the second single condition is not evaluated.

Subquestion 1

From the answer group below, select the correct answer to be inserted in each blank in the following description that concerns the decision condition of the branch coverage used by Company *N*.

Figure 1 shows a sample program to be tested, and Table 1 lists sample test cases for this program. When the program shown in Figure 1 is tested according to the decision condition of the branch coverage used by Company *N* with the test cases, the test result reveals that A in test case 1, and B in test case 2.

- Program (Integer type: *x*, Integer type: *a*, Integer type: *b*, Integer type: *c*, Integer type: *d*)

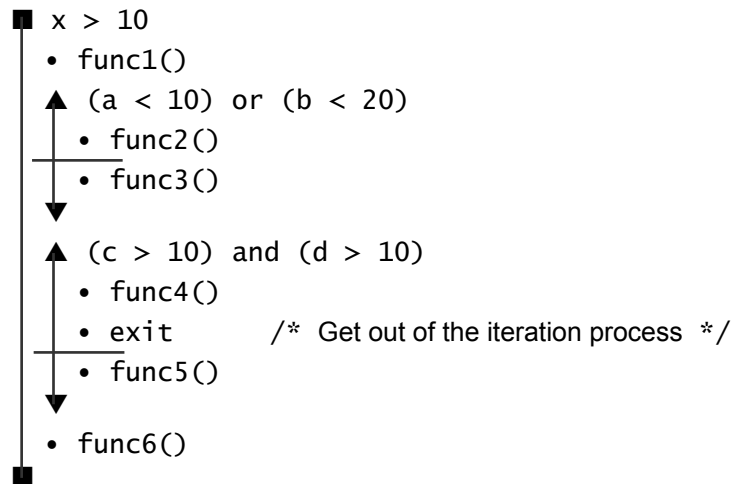


Figure 1 Sample program to be tested

Table 1 Sample test cases

Variable	Test data				
	x	a	b	c	d
Test case 1	11	9	19	10	10
Test case 2	11	10	20	11	11

Answer group for A and B

- a) $b < 20$ is not evaluated
- b) $b < 20$ and $c > 10$ are not evaluated
- c) $b < 20$ and $d > 10$ are not evaluated
- d) $c > 10$ is not evaluated
- e) $c > 10$ and $d > 10$ are not evaluated
- f) $d > 10$ is not evaluated
- g) all single conditions are evaluated

Subquestion 2

The control structure of a program can be described with a control flow graph. In a control flow graph, processes are divided into serial instructions, iteration instructions, and branch instructions, and each of them is placed in a process block (hereinafter, node) that is connected with a directed line segment (hereinafter, edge) in the sequence of process execution. Here, a multiple condition is divided into the respective single conditions, and they are placed in the control flow graph.

Figure 2 is prepared by assigning node numbers (1) through (11) to the sample program to be tested in Figure 1. Figure 3 shows the corresponding control flow graph. The node numbers in Figure 3 correspond to the node numbers in Figure 2. Nodes *S* and *E* in Figure 3 are special nodes that indicate the entry and exit of the program respectively, and there are no corresponding processes in the sample program to be tested.

From the answer group below, select the appropriate node number to be inserted in each blank in the control flow graph in Figure 3.

○ Program (Integer type: x, Integer type: a, Integer type: b,
Integer type: c, Integer type: d)

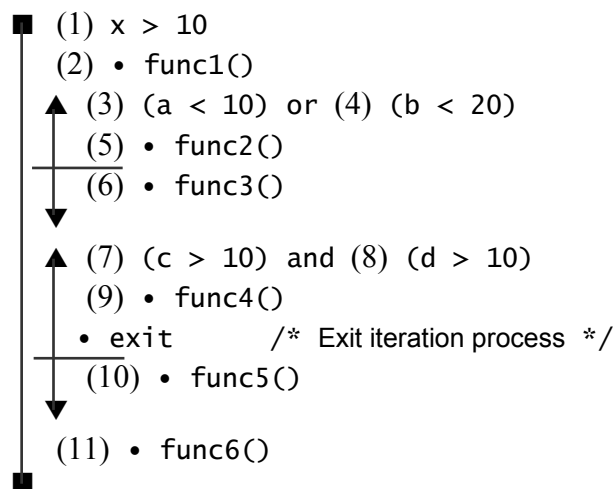
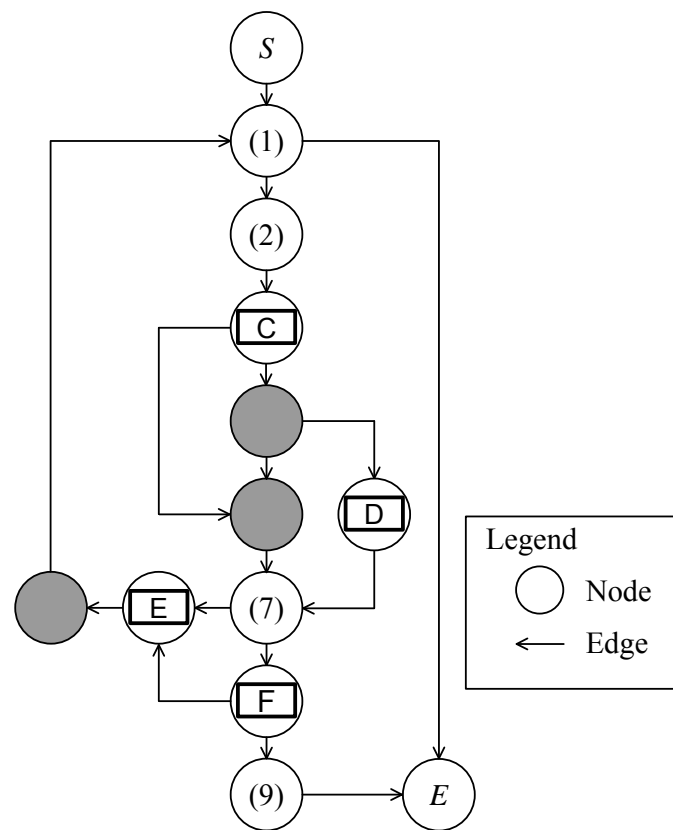


Figure 2 Sample program in Figure 1 with node numbers



Note: Shaded parts are not shown.

Figure 3 Control flow graph that corresponds to sample program in Figure 2

Answer group for C through F

- | | | | |
|--------|---------|---------|--------|
| a) (3) | b) (4) | c) (5) | d) (6) |
| e) (8) | f) (10) | g) (11) | |

Subquestion 3

From the answer group below, select the correct answer to be inserted in each blank in the following description.

For the testing of the program shown in Figure 1, in the case of the branch coverage used by Company *N*, the minimum number of test cases required is G .

Furthermore, there is a method for making test cases by extracting paths from a control flow graph. The minimum number of paths (*S*) that cover all the edges and nodes of a control flow graph is determined with the following expression:

$$S = \text{Number of edges} - \text{Number of nodes} + 2$$

By conducting a test for *S* test cases that correspond to the extracted paths, it is possible to assure higher coverage than branch coverage.

With regard to the concerning the control flow graph in Figure 3, the value of *S* is H . In order to reduce the number of bugs left uncorrected in its programs, Company *N* decides to test the programs with test cases based on control flow graphs.

Answer group for G and H

- | | | |
|------|------|------|
| a) 2 | b) 3 | c) 4 |
| d) 5 | e) 6 | f) 7 |

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

The bin-packing problem is a classic problem in combinatorial optimization. There are items with different sizes, and bins with the same capacity. The problem is to determine the bins of each item, and the items that must be placed into the bin so that the total size of the items in each bin does not exceed the bin capacity, and the number of non-empty bins is minimal. In order to solve the problem, heuristic methods have been developed. Next-Fit, First-Fit, Best-Fit, the most popular methods, programs, and examples are given below.

[Program Description]

- (1) `N` contains the total number of items.
- (2) `C` contains the capacity of the bins. Each bin has the same capacity.
- (3) `A[i]` contains the size of *i*-th item. The size of each item does not exceed the bin capacity `C`. Figure 1 shows an example of the items.
- (4) Indexes of arrays start at 1.
- (5) All of the following three methods place the items in the order in which they arrive.
 - (a) The subprogram `NextFit` places the next item into the current bin if the item fits. If it does not fit, that bin is closed and a new bin is started. Figure 2 shows the execution results of the `NextFit` program with the items shown in Figure 1.
 - (b) The subprogram `FirstFit` places the next item into the lowest numbered bin in which the item fits. If it does not fit into any open bin, a new bin is started. Figure 3 shows the execution results of the `FirstFit` program with the items shown in Figure 1.
 - (c) The subprogram `BestFit` places the next item into the bin that will leave the least capacity left over after the item is placed in the bin. If it does not fit into any open bin, a new bin is started. Figure 4 shows the execution results of the `BestFit` program with the items shown in Figure 1.
- (6) `B[i]` contains the sum of the size of all items placed into *i*-th bin.
- (7) `S[]` is a solution to the problem. `S[i]` contains the bin number into which *i*-th item is placed.
- (8) The subprogram `OutputResult` displays the elements of array `S[]`.

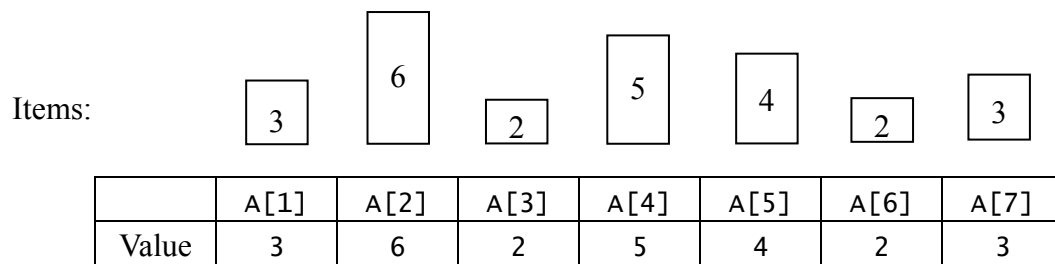


Figure 1 Example of items for programs

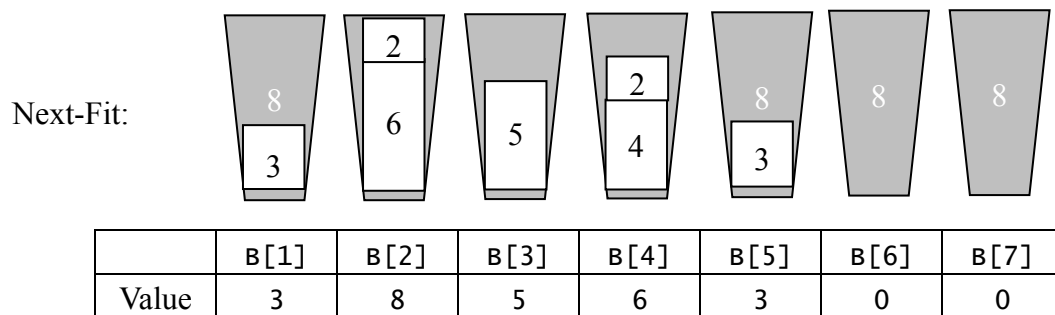


Figure 2 Execution results of NextFit program (C = 8)

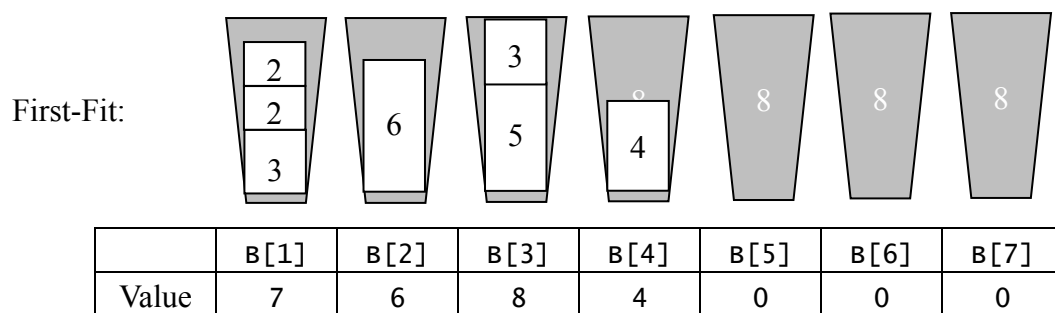


Figure 3 Execution results of FirstFit program (C = 8)

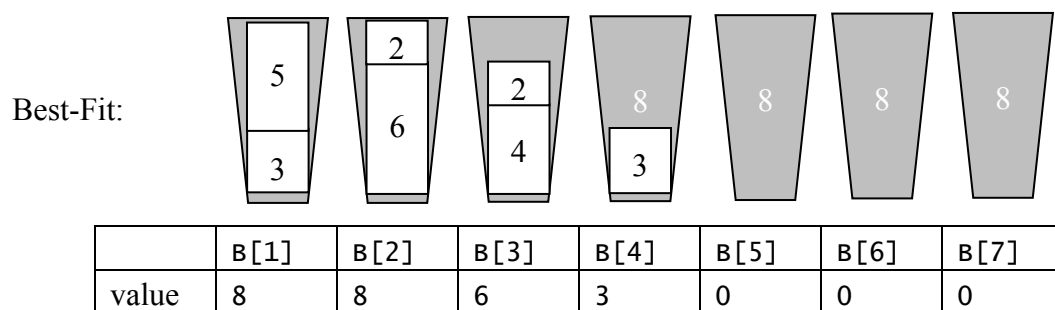


Figure 4 Execution results of BestFit program (C = 8)

[Program]

- Global: Integer type: $N \leftarrow 7, C \leftarrow 8$
- Global: Integer type: $A[N] \leftarrow \{3, 6, 2, 5, 4, 2, 3\}$
- Global: Integer type: $B[N], S[N]$

○ Subprogram: OutputResult()

- Integer type: I

```
■ I: 1, I ≤ N, 1
  • print(S[I]) /* output S[I] */
  • print(" ")  /* output " " */
■
```

○ Subprogram: NextFit()

- Integer type: I, J

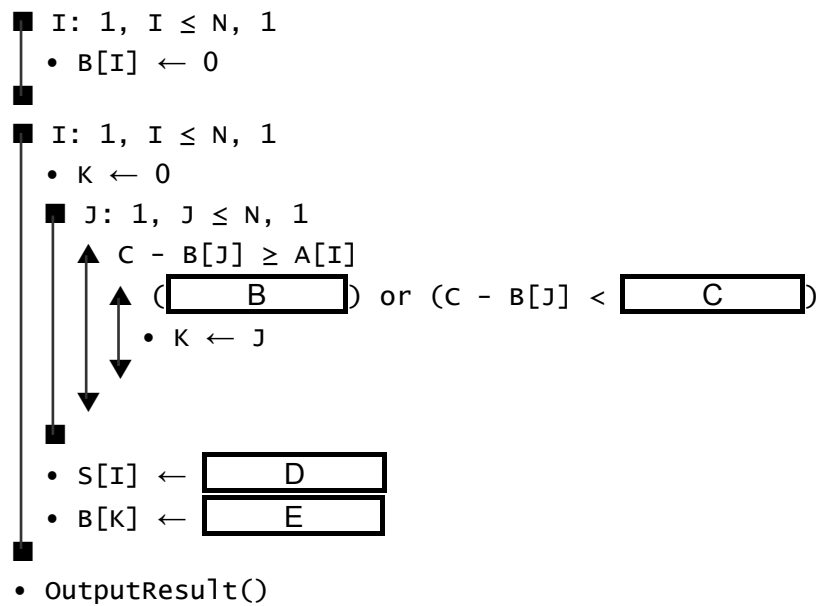
```
■ I: 1, I ≤ N, 1
  • B[I] ← 0
■
  • J ← 1
  ■ I: 1, I ≤ N, 1
    ▲ C - B[J] < A[I]
    │ • J ← J + 1
    ▼
    • S[I] ← J
    • B[J] ← B[J] + A[I]
  ■
    • OutputResult()
```

○ Subprogram: FirstFit()

- Integer type: I, J

```
■ I: 1, I ≤ N, 1
  • B[I] ← 0
■
  ■ I: 1, I ≤ N, 1
    ■ J: 1, J ≤ N, 1
      ▲ C - B[J] ≥ A[I]
      │ • S[I] ← A
      │ • B[J] ← B[J] + A[I]
      │ • J ← N + 1
      ▼
    ■
  ■
    • OutputResult()
```

- Subprogram: BestFit()
- Integer type: I, J, K



Subquestion 1

From the answer groups below, select the correct answer to be inserted in each blank in the above program.

Answer group for A and D

- | | | |
|----------------|-------------|----------------|
| a) A[I] | b) I | c) J |
| d) K | e) S[I] + 1 | f) S[I] + A[I] |
| g) S[I] + B[J] | h) S[I] + J | |

Answer group for B

- | | | |
|----------|----------|----------|
| a) I = J | b) I ≠ J | c) K = 0 |
| d) K = N | e) K > 0 | f) K > J |

Answer group for C

- | | | |
|-------------|-------------|-------------|
| a) A[J] | b) A[K] | c) B[J] |
| d) B[K] | e) C | f) C - A[I] |
| g) C - B[J] | h) C - B[K] | |

Answer group for E

- | | | |
|----------------|----------------|----------------|
| a) B[J] + A[I] | b) B[J] + B[K] | c) B[K] + A[I] |
| d) B[K] + A[J] | | |

Subquestion 2

From the answer group below, select the correct answer to be inserted in each blank in the following description. Here, it is assumed that the correct answers from Subquestion 1 are inserted into all the blanks in the [Program]. If needed, select the same answer twice or more.

In order to simulate another case, the first 3 lines of the program are replaced with the following 3 lines, and then the subprograms are executed.

- Global: Integer type: $N \leftarrow 10, C \leftarrow 10$
- Global: Integer type: $A[N] \leftarrow \{4, 8, 5, 7, 6, 1, 4, 2, 2, 1\}$
- Global: Integer type: $B[N], S[N]$

The following table shows the execution results.

Subprogram	Printed result
NextFit()	<input type="text" value="F"/>
FirstFit()	<input type="text" value="G"/>
BestFit()	<input type="text" value="H"/>

Answer group for F through H

- a) 1 2 1 2 3 4 3 4 4 5
- b) 1 2 1 3 4 1 4 2 3 3
- c) 1 2 1 3 4 3 4 2 3 2
- d) 1 2 3 4 5 5 6 6 6 6
- e) 1 2 3 4 5 6 7 8 9 10
- f) 4 8 5 7 7 9 0 0 0 0
- g) 10 10 10 10 0 0 0 0 0 0

Concerning questions **Q7** and **Q8**, **select one** of the two questions.
Then, mark the **(S)** in the selection area on the answer sheet, and answer the question.
If two questions are selected, only the first question will be graded.

Q7. Read the following description of a C program and the program itself, and then answer Subquestion.

This program is an Armstrong number calculation that will check whether an input number is an Armstrong number. An Armstrong number is an n -digit number that is equal to the sum of the n -th powers of its digits.

For example,

$$371 = 3^3 + 7^3 + 1^3$$

$$1634 = 1^4 + 6^4 + 3^4 + 4^4$$

[Program Description]

- (1) The program receives a number that is a base-10 number.
- (2) The input number is a positive integer. The number of digits of the input number is 8 or fewer, because the sum of the 8-th powers of its digits does not exceed the maximum value of long integer.
- (3) The program calculates the number of digits.
- (4) The program computes the summation of individual digits raised to the power number of digits. If the summation is equal to the input number, the number is an Armstrong number.
- (5) The following 3 functions are used in the program.
 - (i) `long calc_power(int base, int power)`
The function calculates the power number of digits.
 - (ii) `int number_digits(long input_number)`
The function calculates the number of digits.
 - (iii) `int check_armstrong (long n_input)`
The function calculates the summation of the input digits and checks whether the input number is an Armstrong number.
- (6) The following is a sample output of the Armstrong number calculation program.

Enter a number:

8208

8208 has 4 digits.

8208 is an Armstrong number.

[Program]

```
#include <stdio.h>

long calc_power(int, int);
int number_digits(long);
int check_armstrong(long);

int main () {
    long n_input;

    printf("Enter a number:\n");
    scanf("%ld", &n_input);

    printf("%ld has %d digits.\n", n_input, number_digits(n_input));

    if (check_armstrong(n_input))
        printf("%ld is an Armstrong number.\n", n_input);
    else
        printf("%ld is not an Armstrong number.\n", n_input);
    return 0;
}

long calc_power(int base, int power) {
    int i;
    long pw = 1;

    for (i = 0; A; i++)
        pw = pw * base;

    return pw;
}

int number_digits(long input_number) {
    int n_digits = 0;

    while (B) {
        n_digits++;
        C;
    }

    return n_digits;
}
```

```

int check_armstrong(long n_input) {
    long tmp, sum = 0;
    int remainder, digits;

    digits = number_digits(n_input);
    tmp = n_input;

    while (tmp != 0) {
        remainder = tmp % 10;
        sum = sum + ;
        tmp = tmp / 10;
    }

    return ;
}

```

Subquestion

From the answer groups below, select the correct answer to be inserted in each blank in the above program.

Answer group for A

- a) $i < \text{base} + \text{power}$
- b) $i < \text{base} * \text{power}$
- c) $i < \text{power}$
- d) $i \leq \text{base} + \text{power}$
- e) $i \leq \text{base} * \text{power}$
- f) $i \leq \text{power}$

Answer group for B

- a) `input_number != 0`
- b) `input_number != 1`
- c) `input_number == 0`
- d) `input_number == 1`
- e) `n_digits < input_number`
- f) `n_digits <= input_number`

Answer group for C

- a) `input_number = input_number + 10`
- b) `input_number = input_number - 10`
- c) `input_number = input_number * 10`
- d) `input_number = input_number / 10`
- e) `input_number = input_number % 10`

Answer group for D

- a) `calc_power(digits, remainder)`
- b) `calc_power(digits, remainder) * calc_power(n_input, remainder)`
- c) `calc_power(n_input, remainder)`
- d) `calc_power(remainder, digits)`
- e) `calc_power(remainder, n_input)`
- f) `calc_power(remainder, n_input) * calc_power(remainder, digits)`

Answer group for E

- a) `digits != sum`
- b) `digits == sum`
- c) `n_input != sum`
- d) `n_input - sum`
- e) `n_input == sum`
- f) `sum`
- g) `sum - n_input`

Q8. Read the following description of Java programs and the programs themselves, and then answer Subquestion.

[Program Description]

An on-line course gives students badges based on points earned in order to encourage them to study well. To earn points, the students need to take a quiz and answer 10 multiple-choice questions with 4 options (A, B, C and D). Each correct answer is worth 50 points.

GOLD 500	SILVER 300	BRONZE 100
-------------	---------------	---------------

The badges are categorized as GOLD, SILVER and BRONZE with 500, 300, and 100 points, respectively. If the student earns at least 100 points, a BRONZE badge appears, if the points reach at least 300, a SILVER badge appears, and for 500 points, a GOLD badge appears.

The following list shows the execution results of this program.

```
Quiz:
Question 1...
Question 2...
Question 3...
Question 4...
Question 5...
Question 6...
Question 7...
Question 8...
Question 9...
Question 10...
```

```
Student: ID: 123, Name: Thomas Anderson
Answers: A B C A B B C D C A
Badge earned: SILVER 300
```


[Program 1]

```
public A Badge {  
    GOLD(500), SILVER(300), BRONZE(100), NONE(0);  
    private int value;  
  
    private Badge(int value) {  
        this.value = value;  
    }  
  
    public int getValue(){  
        return value;  
    }  
  
    public static Badge getBadge(int score) {  
        for (Badge badge : values()) {  
            if (score >= badge.getValue())  
                return badge;  
        }  
        return NONE;  
    }  
}
```

[Program 2]

```
public class Student {  
    private int id;  
    private String name;  
    private Badge earnedBadge;  
    private char[] answers;  
  
    public Student(int id, String name, B answers) {  
        this.id = id;  
        this.name = name;  
        this.answers = answers;  
    }  
  
    public B getAnswers() {  
        return answers;  
    }  
  
    public void setBadge(Badge earnedBadge) {  
        this.earnedBadge = earnedBadge;  
    }  
}
```

```

public Badge getBadge() {
    return earnedBadge;
}

public String toString() {
    String str = String.format("ID: %s, Name: %s\n Answers: ",
                               id, name);
    for (char answer : answers) {
        str += answer + " ";
    }
    str += String.format("\n Badge earned: %s %d", C);
    return str;
}
}

```

[Program 3]

```

public class Quiz {
    private String[] questions = new String[10];
    private char[] answers = new char[10];

    public Quiz(String[] questions, char[] answers) {
        this.questions = questions;
        this.answers = answers;
    }

    public void setAnswers(char[] answers) {
        this.answers = answers;
    }

    public char[] getAnswers() {
        return answers;
    }

    public String toString() {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < questions.length; i++)
            sb.append(questions[i]).append("\n");
        return sb.toString();
    }
}

```

[Program 4]

```
public class Badges {

    public static int checkQuiz(char[] correctAnswer,
                                char[] studentAnswer ) {
        int score = 0;
        for (int i = 0; i<correctAnswer.length; i++)
            if (correctAnswer[i] == studentAnswer[i])
                D;
        return score;
    }

    public static void main(String[] args) {
        String[] questions = {"Question 1...", "Question 2...",
                               "Question 3...", "Question 4...",
                               "Question 5...", "Question 6...",
                               "Question 7...", "Question 8...",
                               "Question 9...", "Question 10..."};
        char[] answers = {'A', 'B', 'C', 'A', 'D',
                           'B', 'C', 'D', 'D', 'A'};
        Quiz quiz = new Quiz(questions, answers);
        System.out.println("Quiz:\n" + quiz.toString());

        int id = 123;
        String name = "Thomas Anderson";
        char[] studentAnswer = {'A', 'B', 'C', 'A', 'B',
                                'B', 'C', 'D', 'C', 'A'};
        Student student = new Student(id, name, studentAnswer);

        int score = E;
        student.setBadge(Badge.getBadge(score));
        System.out.println("Student: " + student.toString());
    }
}
```

Subquestion

From the answer groups below, select the correct answer to be inserted in each blank in the above programs.

Answer group for A

- a) abstract
- b) class
- c) enum
- d) interface

Answer group for B

- a) Answers
- b) char
- c) char[]
- d) String

Answer group for C

- a) earnedBadge
- b) earnedBadge, earnedBadge.getValue()
- c) earnedBadge.getValue()
- d) earnedBadge.getValue(), earnedBadge

Answer group for D

- a) score *= 10
- b) score *= 50
- c) score += 10
- d) score += 50
- e) score ^= 10
- f) score ^= 50

Answer group for E

- a) checkQuiz(Quiz.getAnswers(), Student.getAnswers())
- b) checkQuiz(Quiz.getAnswers(), student.getAnswers())
- c) checkQuiz(quiz.getAnswers(), Student.getAnswers())
- d) checkQuiz(quiz.getAnswers(), student.getAnswers())