October 2018

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 Ⓢ 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	ⓐ ● ⓒ ⓓ ⓔ ⓕ ⓖ ⓗ ⓘ ⓙ

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
	type: *var1*, ..., *array1*[], ...	Declares variables *var1*, ... , and/or arrays *array1*[], ... , by data *type* such as INT and CHAR.
	FUNCTION: *function*(*type*: *arg1*, ...)	Declares a *function* and its arguments *arg1*, ...
	/* comment */	Describes a comment.
Process	*variable* ← *expression* ;	Assigns the value of the *expression* to the *variable*.
	function (*arg1*, ...) ;	Calls the *function* by passing / receiving the arguments *arg1*, ...

	IF (*condition*) {
	    process1
	}
	ELSE {
	    process2
	}
		Indicates the selection process.
		If the *condition* is true, then *process1* is executed.
		If the *condition* is false, then *process2* is executed, when the optional ELSE clause is present.

	WHILE (*condition*) {
	    process
	}
		Indicates the "WHILE" iteration process.
		While the *condition* is true, the *process* is executed repeatedly.

	DO {
	    process
	} WHILE (*condition*);
		Indicates the "DO-WHILE" iteration process.
		The *process* is executed once, and then while the *condition* is true, the *process* is executed repeatedly.

	FOR (*init* ; *condition* ; *incr*) {
	    process
	}
		Indicates the "FOR" iteration process.
		While the *condition* is true, the *process* is executed repeatedly.
		At the start of the first iteration, the process *init* is executed before testing the *condition*.
		At the end of each iteration, the process *incr* is executed before testing the *condition*.

[Logical constants]

true, false

[Operators and their priorities]

Type of operation	Unary	Arithmetic		Relational	Logical	
Operators	+, −, not	×, ÷, %	+, −	>, <, ≥, ≤, =, ≠	and	or
Precedence	High ←				→ Low	

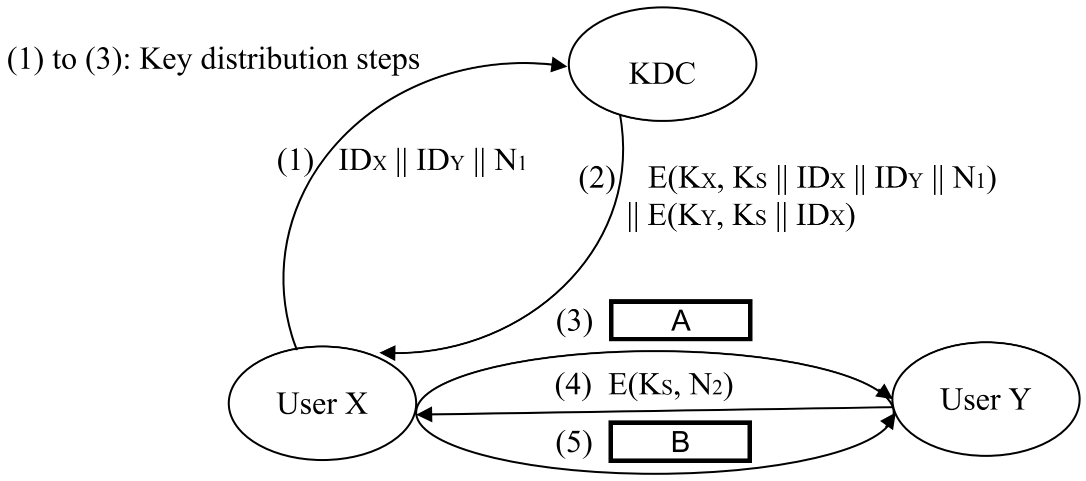
Note: With division of integers, an integer quotient is returned as a result.
The "%" operator indicates a remainder operation.

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

Q1. Read the following description of symmetric key encryption, and then answer Subquestions 1 and 2.

When a user wants to communicate messages with other user, they first set up a session and then use an encryption technique to protect the message from intruders. In symmetric key encryption, both users share the same key to exchange information.
A key distribution center (hereinafter, KDC) delivers a unique master key to the concerned users in some non-cryptographic manner such as physical delivery. Hence, the master key is known to the KDC and the concerned users only. For each communication between the users, the message is encrypted using a temporary key, referred to a session key, obtained from the KDC. The session key is transmitted to the users in encrypted form using the master key provided by the KDC.

Figure 1 shows a typical key distribution and authentication steps, when user X, with master key K_X, wishes to establish a logical connection with user Y, with master key K_Y.


(1) to (3): Key distribution steps
KDC
(1) IDX || IDY || N1
(2) E(KX, KS || IDX || IDY || N1) || E(KY, KS || IDX)
(3) A
User X
(4) E(KS, N2)
User Y
(5) B
(3) to (5): Authentication steps

Symbols: $E(k, m)$: encryption function that encrypts the message m using the key k
|| : Concatenation operator

ID_X: Identity (network address) of X	ID_Y: Identity (network address) of Y
K_X: Master key for X	K_Y: Master key for Y
N_1, N_2: Nonces (transaction identifiers)	K_S: Session key for session S

Figure 1 Key distribution and authentication steps

The following description explains steps (1) to (5) in Figure 1. Note that steps (1) to (3) are key distribution steps, and steps (3) to (5) are authentication steps.

(1) User X issues a session key request to the KDC. The message sent to the KDC includes identities (network addresses) of X and Y, represented by ID_X and ID_Y, and a unique identifier N_1 for this transaction, known as a nonce. A random number is a good choice for a nonce, such that it is difficult for an opponent to guess.

(2) The KDC responds with a message to user X. The message has two parts: the first part is for user X and the second part is for user Y.
The first part of the message is encrypted using K_X, such that only user X can read the message. This message contains the one-time session key K_S to be used for the session, and the original message including the nonce to enable user X to match this response with the message sent in step (1).
The second part of the message is encrypted using K_Y, such that only user Y can read the message. This message contains the session key K_S and ID_X.

(3) User X stores the session key K_S for use in the upcoming session, and forwards a part of the message received in step (2) to user Y. At this time, the session key has been securely delivered to users X and Y, and user Y knows that user X wants to communicate with the session key K_S.

(4) Using the newly established session key for encryption, user Y sends a nonce N_2 to user X.

(5) User X responds with an encrypted message. This message contains a transformed value of N_2. Here, f(*n*) is a function that performs some transformation (e.g., adding one) on *n*. By this step, user Y ensures that user X, has the session key for this session.

Subquestion 1

From the answer group below, select the correct answer to be inserted in each blank $\boxed{}$ in Figure 1.

Answer group

a) $E(K_S, f(N_2))$	b) $E(K_S, N_2)$
c) $E(K_X, f(N_2))$	d) $E(K_X, K_S \| ID_X)$
e) $E(K_Y, K_S \| ID_X)$	f) $E(K_Y, K_S \| ID_Y)$

Subquestion 2

From the answer groups below, select the appropriate answer to be inserted in each blank $\boxed{}$ in the following description.

The KDC supplies $\boxed{\text{C}}$ to users in advance. When user X wants to communicate messages with user Y, the KDC generates $\boxed{\text{D}}$ before starting communications between the users.

When user X issues a request to the KDC for a session key, the KDC sends back the original message along with the requested session key. The main reasons why the original message including the nonce is sent back to user X are $\boxed{\text{E}}$ and $\boxed{\text{F}}$.

Answer group for C and D

a) master key(s)	b) network address(es)
c) session key(s)	d) transaction identifier(s)

Answer group for E and F

a) user X can confirm that the session key has not been compromised
b) user X can know that it originated at the KDC
c) user X can send the encrypted original message to user Y
d) user X can use this response as a digital signature
e) user X can verify that the request has not been altered
f) user X can verify that this response is not a replay attack based on some previous sessions

Q2. Read the following description of paging, and then answer Subquestions 1 and 2.

Paging is a memory-management scheme that maps logical addresses onto physical addresses.
In a system that manages its memory by paging, logical address space is broken into blocks called pages. Physical address space is also broken into blocks called frames. Pages are usually placed on an auxiliary storage and loaded into available frames on demand.
A logical address consists of two parts: a page number *p* and displacement *d*. Similarly, a physical address consists of a frame number *f* and displacement *d*. The size of *d* is called the page size, which is fixed and common to both address spaces.
The page number is used as an index into a page table. The page table maintains the correspondence between the page number and the frame number. By using the page table, a logical address can be translated into physical address by replacing the page number with its corresponding frame number.

Subquestion 1

From the answer groups below, select the correct answer to be inserted in each blank $\boxed{\qquad}$ in the following description.

Figure 1 shows an example of specification of logical and physical addresses. The logical address is 24-bit long, which covers 16M (2^{24}) addresses, and the physical address is 18-bit long, covering 256k (2^{18}) addresses. The page size is 12-bit long, covering 4,096 ($=2^{12}$) addresses.

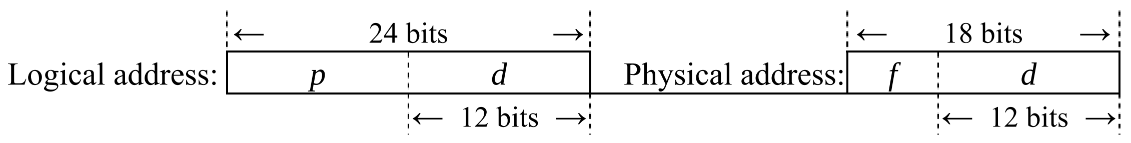


Figure 1 Example of specification of logical and physical addresses

In Figure 1, the number of pages is 4,096, and the number of frames is $\boxed{\text{ A }}$.
Generally, when the logical address is *n*-bit long, and the page size is *m*-bit long, the number of pages is expressed as $\boxed{\text{ B }}$.

Answer group for A

a) 16	b) 64	c) 256	d) 4,096

Answer group for B

a) 2^m	b) 2^{n-m}	c) m^2	d) $n - m$

Subquestion 2

From the answer groups below, select the correct answer to be inserted in each blank $\boxed{\qquad\quad}$ in the following description.

There is a system that features a paging mechanism. The specification of logical and physical addresses is as shown in Figure 1.
Figure 2 illustrates a situation in which the pages in the logical address space are mapped onto the frames in the physical address space according to the page table at the moment. At this moment, the page table shows that the data at logical address 006C00 is $\boxed{\quad\text{C}\quad}$, and the data at logical address 007E00 is $\boxed{\quad\text{D}\quad}$. In this virtual storage system that uses the page table shown in Figure 2, $\boxed{\quad\text{E}\quad}$ and then the system starts I/O operations to load the required page into the physical address space. The addresses are shown in hexadecimal.

Page Table:

Page No.	L	Frame No.
0	1	0
1	1	1
2	1	2
3	1	3
4	0	–
5	1	7
6	0	–
7	1	6
…	…	…

Note:
L indicates the page loading status.
- if L is 1, the page is loaded into the frame.
- if L is 0, the page is not loaded.

Figure 2 Contents of the page table (partial)

Answer group for C and D

a) located at physical address 05E00
b) located at physical address 06C00
c) located at physical address 06E00
d) located at physical address 07C00
e) not loaded into the physical address space

Answer group for E

a) a page fault occurs when the requested page's L bit is 0
b) a page fault occurs when the requested page's L bit is 1
c) an I/O interruption occurs when the requested page's L bit is 0
d) an I/O interruption occurs when the requested page's L bit is 1

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

Company U is a painting company that offers interior and exterior painting services depending on the customer's contract. Company U provides services according to the customer's needs. Some customers need to paint the whole building, and some only need to paint a bathroom, kitchen or bedroom. Some customers need water resistant color, and some need 3-year guarantee color.
Company U provides high-quality painting colors from its suppliers. A particular type of painting color is obtained from a designated supplier.
Company U's staff work on a per-hour basis. A customer can contract one or more jobs, and one or more staff members are assigned to each job.

The database uses five tables. The table structure and sample data of each table is shown below. Here, underline (___) indicates the primary key, and dotted underline (.....) indicates the foreign key.

(1) Customer table
The Customer table contains information about the customers.

<u>CustomerID</u>	CustomerName	CustomerAddress
1	Maung Maung	Yangon
2	Aung Aung	Mandalay
3	Hla Hla	Bagan

(2) Staff table
The Staff table contains information associated with the staffs. Rate indicates the payment by the hour.

<u>StaffID</u>	StaffName	Rank	Rate
1	Kyaw Kyaw	In charge	80
2	Zaw Zaw	Painter	50
3	Min Min	Painter	50

(3) Supplier table
The Supplier table contains information about the suppliers.

<u>SupplierID</u>	SupplierName	PaintQuality	UnitPrice
1	Daw Saw	Water Resistant	100
2	U Win	Oil Resistant	200
3	Daw Thein	3-Year Guarantee	300

(4) Job table

The Job table contains information about the contracted jobs. The total surface area is in units of square meters.

JobID	TotalSurfaceArea	CustomerID	SupplierID
1	100	2	3
2	10	1	1
3	20	1	2
4	30	3	3
5	15	3	1

(5) JobStaff table

The JobStaff table is a joint table between the Job table and Staff table. A job can have many staff members and a staff member can do many jobs.

JobStaffID	JobID	StaffID	WorkingHours
1	1	1	2
2	1	2	4
3	1	3	4
4	2	1	5
5	3	2	8
6	4	3	5
7	5	2	8

Subquestion 1

From the answer groups below, select the correct answer to be inserted in each blank $\boxed{}$ in the following SQL statement.

The SQL statement named SQL1 outputs the staff names and their total number of jobs for the staff members who have more than two jobs.

```
-- SQL1 --
SELECT S.StaffID, S.StaffName, [    A    ] AS TotalJob
FROM Job J, Staff S, JobStaff JS
WHERE J.JobID = JS.JobID AND S.StaffID = JS.StaffID
[    B    ]
```

From the sample data of each table shown in the description, SQL1 outputs the following result.

StaffID	StaffName	TotalJob
2	Zaw Zaw	3

Answer group for A

a) `AVG(J.TotalSurfaceArea)` b) `COUNT(*)`

c) `J.JobID` d) `SUM(*)`

Answer group for B

a) `GROUP BY S.StaffID, S.StaffName`

b) `GROUP BY S.StaffID, S.StaffName HAVING COUNT(*) > 2`

c) `HAVING COUNT(*) > 2`

d) `HAVING COUNT(*) >= 2 GROUP BY S.StaffID, S.StaffName`

Subquestion 2

From the answer group below, select the correct answer to be inserted in the blank $\boxed{}$ in the following SQL statement.

The SQL statement named SQL2 outputs the job IDs and their total surface areas where the customers need "3-Year Guarantee" color.

```
-- SQL2 --
SELECT J.JobID, J.TotalSurfaceArea
FROM Job J, Supplier S
[    C    ] J.SupplierID = S.SupplierID
        AND S.PaintQuality = '3-Year Guarantee'
```

From the sample data of each table, SQL2 outputs the following result.

JobID	TotalSurfaceArea
1	100
4	30

Answer group for C

a) `GROUP BY` b) `HAVING` c) `ORDER BY` d) `WHERE`

Subquestion 3

From the answer group below, select the correct answer to be inserted in each blank $\boxed{}$ in the following SQL statement.

The SQL statement named SQL3 outputs the total surface area and total labor costs for each job.

```
-- SQL3 --
SELECT JobID, TotalSurfaceArea, [    D    ] AS TotalLaborCosts
FROM (SELECT JS.JobID AS JobID,
             J.TotalSurfaceArea, [    E    ] AS LaborCosts
      FROM Job J, Staff S, JobStaff JS
      WHERE JS.StaffID = S.StaffID AND JS.JobID = J.JobID) LC
GROUP BY JobID, TotalSurfaceArea
```

From the sample data of each table, SQL3 outputs the following result.

JobID	TotalSurfaceArea	TotalLaborCosts
1	100	560
2	10	400
3	20	400
4	30	250
5	15	400

Answer group for D and E

a) `LC.LaborCosts`
b) `SUM(LC.LaborCosts)`
c) `SUM(WorkingHours * S.Rate)`
d) `SUM(WorkingHours) * S.Rate`
e) `WorkingHours * S.Rate`

Q4. Read the following description of a network in a university, and then answer Subquestions 1 and 2.

In order to improve security, a university network is divided into two segments: the Demilitarized Zone (DMZ) and Local Area Network (LAN). The LAN is the internal part of the network for storing all types of data. Any external access to the university information is made through the DMZ. The network administrator has implemented a number of security policies on the firewall according to the university's security standards, such that data to be made public must be transferred to the DMZ at first. A router can perform the firewall functionalities. Figure 1 shows the network configuration of the university.

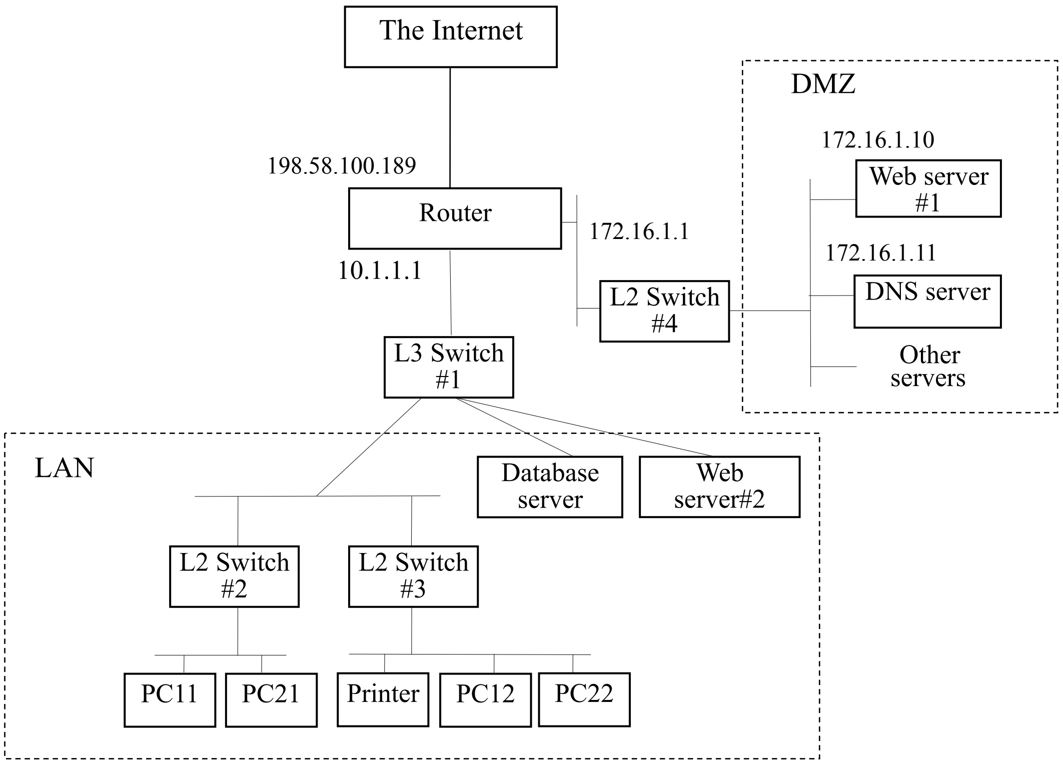


Figure 1 Network configuration of the university

Subquestion 1

From the answer group below, select the correct answer to be inserted in each blank $\boxed{}$ in Table 1.

In order to secure the servers located in the DMZ, the network administrator implements the traffic rules shown in Table 1 on the router according to the following rules:

(1) Secure Web services on the external IP address of the router will be mapped to Web server #1 in the DMZ.

(2) All of the servers can access the Internet from the DMZ using NAT (Network Address Translation).

(3) Render all of the servers accessible from the network administrator in the LAN.

(4) Even if any of the servers in the DMZ are attacked, it will be impossible for the attackers to reach devices located in the LAN.

Here, the DMZ uses the subnet 172.16.1.0/26.

Table 1 Traffic rules on the router (other settings are not shown)

Source	Destination	Services	Action	Translation
the Internet	$\boxed{\text{A}}$	https	$\boxed{\text{B}}$	Map 172.16.1.10
DMZ	the Internet	$\boxed{\text{C}}$	Allow	NAT
$\boxed{\text{D}}$	DMZ	ssh	Allow	-

Answer group

a) 10.1.1.1	b) 172.16.1.1	c) 172.16.1.10	d) 198.58.100.189
e) Allow	f) Any	g) Deny	h) https
i) LAN	j) ssh		

Subquestion 2

From the answer groups below, select the correct answer to be inserted in each blank $\boxed{}$ in the following description.

In order to improve network security, the network administrator is planning to implement two VLANs by segmenting the existing LAN without physically rearranging the devices. One VLAN named VLAN 10 is for students, and the other named VLAN 20 is for teachers. All the switches have the VLAN spanning feature, and all the devices placed under the same VLAN can access each other across the different switches.

Figure 2 shows the re-designed network configuration of the university. Note that L3 switch #1 is accessible from all VLANs.

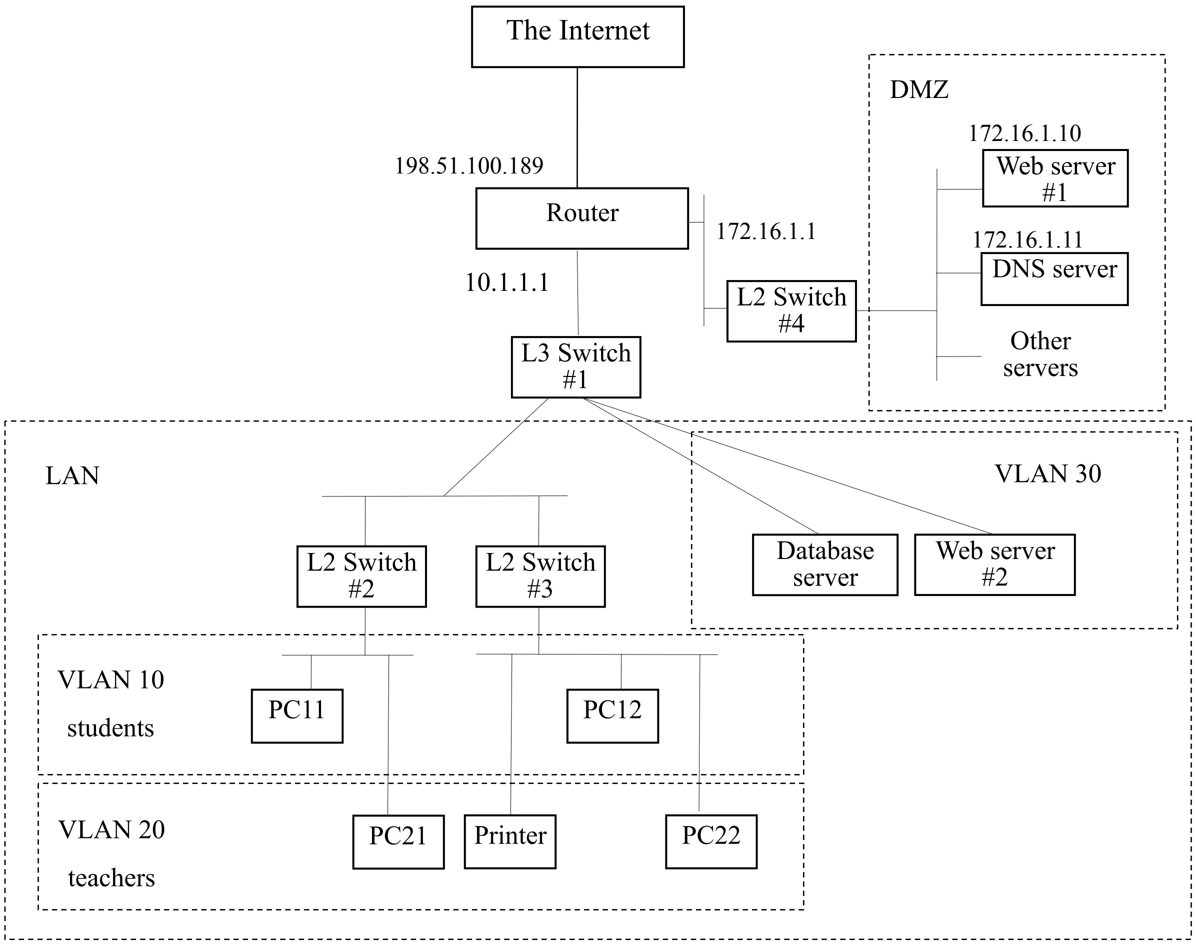


Figure 2 Re-designed network configuration of the university

When PC21 in VLAN 20 tries to connect to a Web site `https://www.info.sample.edu` that is running on Web server #1, PC21 sends two messages before sending an http request message.

(1) PC21 sends a broadcast message in order to obtain the MAC address of L3 Switch #1.

(2) PC21 sends a query message in the local network in order to resolve the URL.

Message (1) can be seen by only the devices $\boxed{\quad E \quad}$.

Message (2) is sent for $\boxed{\quad F \quad}$.

Answer group for E

a) in DMZ
b) in VLAN 20
c) that are connected to L2 Switch #2
d) that are connected to L2 Switch #2 and L2 Switch #3

Answer group for F

a) Database server	b) DNS server
c) Web server #1	d) Web server #2

Q5. Read the following description of a training management system, and then answer Subquestion.

Company W operates an IT training center. Company W utilizes a training management system (hereinafter, the system) to manage students and training courses.
Any person can view the IT training information via the Web site. If the person wants to attend the IT training courses, he or she must register the student information to the system. After registration, the person will receive a username and password by e-mail from the system. After logging into the system by using the username and password, the person needs to download the application form, fill in the required information, and upload the application form for course enrollment.
Some courses have multiple classes: parallel classes such as Monday class and Friday class, and seasonal classes such as winter class and spring class.
Figure 1 shows the use case diagram of the system.

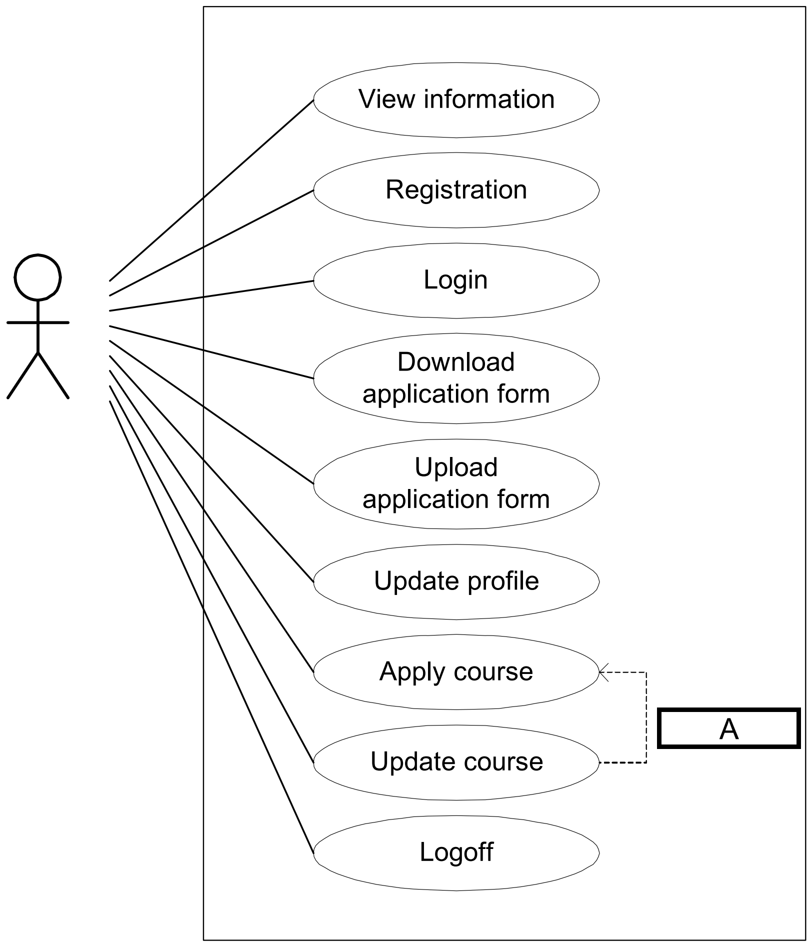


Figure 1 Use case diagram of the system

Table 1 shows the description of each use case shown in Figure 1.

Table 1 Description of each use case shown in Figure 1

No.	Description
01	View information A student can view all information associated with IT Training.
02	Registration If a student would like to apply for IT training courses, he or she needs to register the student information to the system. The system then issues a username and password.
03	Login If a student wants to update the profile, apply for courses, or update the courses, he or she must login to the system.
11	Download application form If a student wants to apply for courses, he or she must download an application form.
12	Upload application form After filling in the necessary information, the student must upload the application form. After uploading, the student must confirm the course application.
13	Update profile A student can update the registered student information.
14	Apply course A student must confirm the course application for which he or she wants to apply. After applying for the course, the student will receive the course information by e-mail.
15	Update course A student can cancel the applied course, or change the class of the applied course. After updating the course, the student will receive the course update notice by e-mail.
16	Logoff Logs off the system.

Figure 2 shows the screen transition diagram of the system.
In Figure 2, P*nn* indicates a screen page. Except for P00 and P10, *nn* corresponds to the use case number in Table 1. EM*m* indicates an e-mail message.

Subquestion

From the answer groups below, select the appropriate answer to be inserted in each blank $\boxed{}$ in Figure 1 and Figure 2.

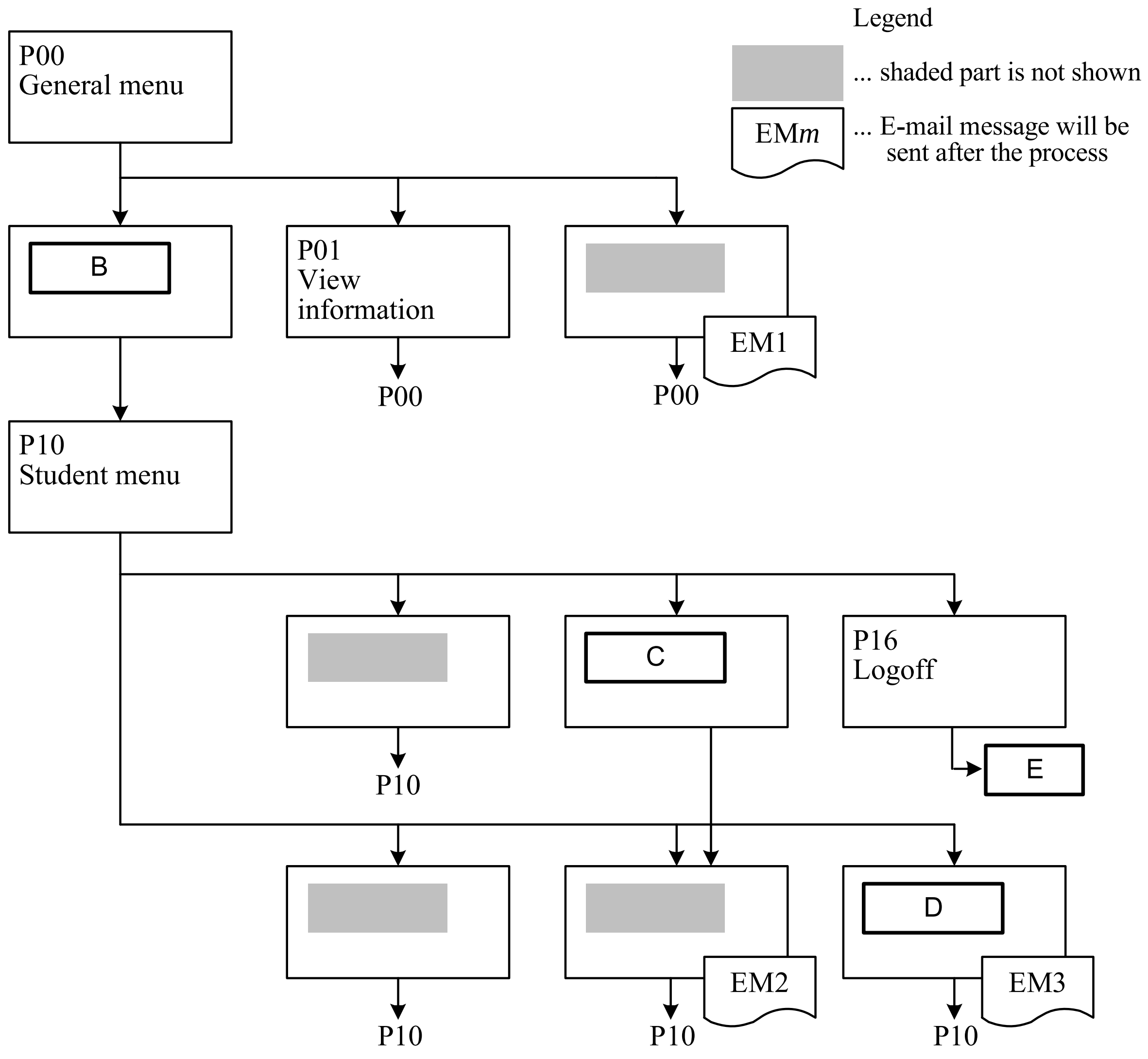


Figure 2 Screen transition diagram of the system

Answer group for A

a) << extend >> b) << include >>

Answer group for B through D

a) P02 Registration b) P03 Login

c) P11 Download application form d) P12 Upload application form

e) P13 Update profile f) P14 Apply course

g) P15 Update course

Answer group for E

a) either P01 or P10 b) P00

c) P02 d) P10

Q6. Read the following description of programs and the programs themselves, and then answer Subquestions 1 and 2.
(See the top of this booklet for the revised notations used for pseudo-language.)

[Description of Program 1]

There are N persons (numbered from 1 to N) to attend a party. At the end of the party, M cakes are prepared as gifts for the persons. The i-th cake has weight W[i] and is presented to person P[i]. Each person is gifted at least one cake.

Before leaving the party, persons are welcome to exchange their cakes with others. That is, person A can give one cake to person B and get another cake back from person B. An exchange is called "friendly" if the difference between the weights of the gifts owned by each of the two persons is reduced after the exchange. The subprogram Exchange is to output possible friendly exchanges between the guests.

(1) Table 1 shows an example of gift arrangement. Table 1 indicates that:
- The 1st person (indicated by P[4]) received the 4th cake, of which the weight is 8
- The 2nd person (indicated by P[1] and P[3]) received the 1st and 3rd cakes, of which the weights are 10 and 6, respectively
- The 3rd person (indicated by P[2]) received the 2nd cake, of which the weight is 3
- The 4th person (indicated by P[5]) received the 5th cake, of which the weight is 9

Table 1 Example of gift arrangement

i	*1*	*2*	*3*	*4*	*5*
Cake's Weight (W[i])	10	3	6	8	9
Person received (P[i])	2	3	2	1	4

(2) Regarding the data shown in Table 1, there are 4 friendly cake exchanges.
The following list shows the output of function Exchange(4, 5, W[], P[]) with parameters W[] and P[] shown Table 1.

```
Exchange between cake 1 and 2
Exchange between cake 1 and 4
Exchange between cake 1 and 5
Exchange between cake 3 and 2
```

(3) Assume that there is an exchange of a cake with weight x owned by a person whose gift's total weight is T_x for a cake with weight y owned by a person whose gift's total weight is T_y. If $T_x > T_y$ and it is a friendly exchange, then:

$$T_x - T_y > |(T_x - x + y) - (T_y - y + x)| \Rightarrow x - y > 0 \text{ and } T_x - T_y > x - y \qquad (\alpha)$$

(4) The argument specification for subprogram Exchange is given in Table 2.

Table 2 Argument specification for subprogram Exchange

Variable	Input/Output	Description
N	Input	The number of persons who attend the party
M	Input	The number of cakes given to persons (M ≥ N)
W[] P[]	Input	Two arrays contain M elements each, where a tuple W[i], P[i] indicates the weight of the i-th cake and the person who receives that cake (1 ≤ P[i] ≤ N)

(5) Subprogram Print is defined to output sentences. For example, Print("Exchange between cake ", I, " and ", J) outputs "Exchange between cake 1 and 2" if variable I is 1 and variable J is 2.

Subprogram Exchange calculates the total weight of each person's cakes first, then stores them into array T. Based on this, it outputs all combinations of two cakes that satisfy equation (α). The indexes of arrays start at 1.

[Program 1]

```
FUNCTION: Exchange(INT: N, INT: M, INT: W[], INT: P[]) {
   INT: T[], I, J

   FOR (I ← 1; I ≤ N; I ← I + 1) {
      T[I] ← [   A   ];
   }
   FOR (I ← 1; I ≤ M; I ← I + 1) {
      T[P[I]] ← [   B   ];
   }
   FOR (I ← 1; I ≤ M; I ← I + 1) {
      FOR (J ← 1; J ≤ M; J ← J + 1) {
         IF ((T[P[I]] > T[P[J]]) [   C1   ]
             (W[I] > W[J]) [   C2   ]
             ((T[P[I]] - T[P[J]]) > (W[I] - W[J]))) {
            Print("Exchange between cake ", I, " and ", J);
         }
      }
   }
}
```

Subquestion 1

From the answer groups below, select the correct answer to be inserted in each blank $\boxed{}$ in Program 1. Here, the answers to be inserted in C1 and C2 should be selected as an appropriate combination from the answer group for C.

Answer group for A

a) `0` b) `P[I]` c) `P[W[I]]`

d) `T[P[I]]` e) `W[I]` f) `W[P[I]]`

Answer group for B

a) `T[I] + W[I]` b) `T[I] + W[P[I]]`

c) `T[P[I]] + W[I]` d) `T[P[I]] + W[P[I]]`

e) `W[I]` f) `W[P[I]]`

Answer group for C

	C1	C2
a)	and	and
b)	and	or
c)	or	and
d)	or	or

Subquestion 2

From the answer groups below, select the correct answer to be inserted in each blank $\boxed{}$ in the following description. Here, the answers to be inserted in F1, F2 and F3 should be selected as an appropriate combination from the answer group for F.

[Description of Program 2]

Subprogram `RepeatExchange` exchanges cakes again and again based on the rule mentioned above. It outputs the sequence of exchanges and total weight of cakes of each person when all sequential exchanges have finished. The argument specification for subprogram `RepeatExchange` is the same as subprogram `Exchange`. Here, assume that the correct answers are inserted in blanks $\boxed{\text{ A }}$ through $\boxed{\text{ C2 }}$ in Program 2.

[Program 2]

```
FUNCTION: RepeatExchange(INT: N, INT: M, INT: W[], INT: P[]) {
   INT: T[], I, J, FLAG, TEMP

   FLAG ← 0;
   FOR (I ← 1; I ≤ N; I ← I + 1) {
      T[I] ← [   A   ];
   }
   FOR (I ← 1; I ≤ M; I ← I + 1) {
      T[P[I]] ← [   B   ];
   }
   WHILE (FLAG = 0) {
      FLAG ← 1;
      FOR (I ← 1; I ≤ M; I ← I + 1) {
         FOR (J ← 1; J ≤ M; J ← J + 1) {
            IF ((T[P[I]] > T[P[J]]) [   C1   ]
                (W[I] > W[J]) [   C2   ]
                ((T[P[I]] - T[P[J]]) > (W[I] - W[J]))) {
               Print("Exchange between cake ", I, " and ", J);
               T[P[I]] ← T[P[I]] - W[I] + W[J];
               T[P[J]] ← T[P[J]] - W[J] + W[I];
               TEMP ← P[I];
               P[I] ← P[J];
               P[J] ← TEMP;
               FLAG ← 0;
            }
         }
      }
   }
   FOR (I ← 1; I ≤ N; I ← I + 1) {
      Print("Total weight of person ", I, " is ", T[I]);
   }
}
```

Table 3 shows another example of gift arrangement.

Table 3 Another example of gift arrangement

i	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
W[i]	15	12	9	8	6	1	1	1	1
P[i]	1	1	1	2	2	2	3	3	3

The following list shows the output of function `RepeatExchange(3, 9, W[], P[])` with parameters `W[]` and `P[]` shown Table 3.

```
Exchange between cake 1 and 4
Exchange between cake 1 and 7
Exchange between cake [   D   ]
Exchange between cake 3 and 6
Exchange between cake [   E   ]
Total weight of person 1 is [   F1   ]
Total weight of person 2 is [   F2   ]
Total weight of person 3 is [   F3   ]
```

Answer group for D and E

a) `1 and 8`   b) `2 and 4`   c) `2 and 5`   d) `2 and 6`
e) `2 and 7`   f) `3 and 4`   g) `3 and 5`   h) `4 and 6`
i) `4 and 7`   j) `7 and 8`

Answer group for F

	F1	F2	F3
a)	16	18	20
b)	17	18	19
c)	17	19	18
d)	18	17	19
e)	18	18	18
f)	18	19	17
g)	19	17	18
h)	19	18	17

Concerning questions **Q7** and **Q8**, **select one** of the two questions.
Then, mark the Ⓢ 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 Subquestions 1 and 2.

Caesar's cipher is a simple cryptography that uses a substitution technique. Given a message (a text to be encrypted) and a key (an integer number *k*), each character in the message is replaced by the character *k* positions down the alphabet.
For example, given "abz" as a message and integer 3 as a key, the character "a" in the message is replaced by the character "d" three positions down the alphabet (a→b→c→d). Similarly, "b" is replaced by "e" (b→c→d→e), and "z" is replaced by "c" (z→a→b→c). In this cipher, the alphabets wraps around and "a", "b", … follows "z". Therefore, the message "abz" is encrypted as "dec". Figure 1 shows an example of encryption using Caesar's cipher.

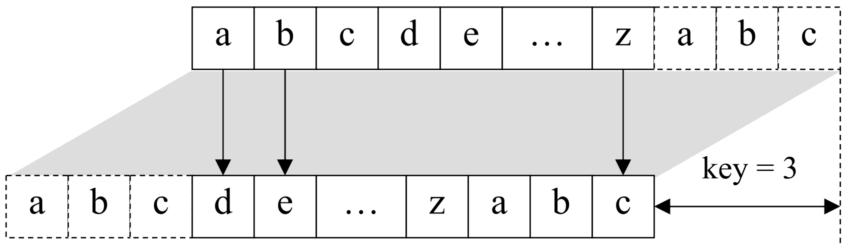


Figure 1 Example of the encryption using Caesar's cipher

[Program Description]
(1) The program reads two inputs: a message and a key.
  (i) A message which is a character string whose elements are only alphabetic characters (26 upper-case letters "A" to "Z" and 26 lower-case letters "a" to "z"). The length of a message is between 1 and 100.
  (ii) A key which is an integer number between 1 and 25.
(2) The program consists of the following functions.
  (i) The function `encrypt` encrypts the given message by Caesar's cipher using the given key. It returns the encrypted message by reference.
  (ii) The function `is_alpha` checks whether the given message contains only alphabets. It returns 1 if the message contains only alphabets, and 0 if it does not.
  (iii) The function `main` reads two inputs from the standard input, calls the function `encrypt`, and outputs the encrypted message to the standard output. If the message contains any non-alphabetic characters, it prompts for another message. If the key

is less than 1 or greater than 25, it prompts for another key.

(3) The following is a sample output of the program.

```
Enter a message to encrypt: abz89
Enter a message to encrypt: AbZ
Enter key: 0
Enter key: 3
Encrypted message: DeC
```

[Program]

```
#include <stdio.h>

#define MAX_LENGTH (100)

void encrypt(char [    A    ], int key) {
   char ch;
   int i;

   for (i = 0; message[i] != '\0' && message[i] != '\n' &&
               message[i] != '\r'; i++) {
      ch = message[i];
      if (ch >= 'a' && ch <= 'z') {
         ch = [    B    ];
      }
      else {
         ch = [    C    ];
      }
      message[i] = ch;
   }
}

int is_alpha(const char* message) {
   char ch;
   int i;

   for (i = 0; message[i] != '\0' && message[i] != '\n' &&
               message[i] != '\r'; i++) {
      ch = message[i];
      if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <='Z')) {
         continue;
      }
      return 0;
   }
   return i != 0;
}
```

```
int main() {
   char message[MAX_LENGTH + 2];
   int key;

   do {
      printf("Enter a message to encrypt: ");
      fgets(message, MAX_LENGTH, stdin);
   } while ([    D    ]);

   do {
      printf("Enter a key: ");
      if (scanf("%d", &key) == 0) {
         continue;
      }
   } while ([    E    ]);         /* α */

   encrypt(message, key);
   printf("Encrypted message: %s\n", message);
   return 0;
}
```

Subquestion 1

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

Answer group for A

a) `&message`
b) `**message`
c) `*message`
d) `message`

Answer group for B and C

a) `'A' + (ch + key) % 26`
b) `'A' + (ch - 'A' + key) % 26`
c) `'a' + (ch + key) % 26`
d) `'a' + (ch - 'a' + key) % 26`
e) `(ch + key) % 26`
f) `(ch - key) % 26`
g) `ch + key`
h) `ch - key`

Answer group for D and E

a) `!is_alpha(*message)`
b) `!is_alpha(message)`
c) `is_alpha(*message)`
d) `is_alpha(message)`
e) `key < 1 || key > 25`
f) `key <= 1 || key >= 25`
g) `key > 1 && key < 25`
h) `key >= 1 && key <= 25`

Subquestion 2

From the answer groups below, select the correct answer to be inserted in each blank $\boxed{}$ in the following description.

The program converts the message “abz” to “dec” with the key value 3. The program is to be changed to accept a negative key value, such that the (encrypted) message “dec” can be converted to the (original) message “abz” with key value -3.

In order to accept a negative key value and perform the function described above, the program is changed as follows:

(1) Expand the range of key value to $-25 \leq$ key value ≤ 25, including 0. Encrypted message is equal to original if 0 is given to key value.

(2) Replace the following line marked `/* α */` in the function `main`:

```
} while (    E    );          /* α */
```

with the following 3 lines:

```
} while (    F    );
if (key < 0)
      G    ;
```

Here, the result of *k* % *d* (*d* > 0) has the same sign as *k*. For example, -12 % 5 is -2.

Answer group for F

a) `key != (key % 25)`	b) `key != (key % 26)`
c) `key == (key % 25)`	d) `key == (key % 26)`

Answer group for G

a) `key = key % 25`	b) `key = key % 26`
c) `key = 25 - key`	d) `key = 26 - key`
e) `key += 25`	f) `key += 26`

Q8. Read the following description of Java programs and the programs themselves, and then answer Subquestions 1 through 3.

[Program Description]

The programs implement arithmetic operations in double numbers and rational numbers, consisting of one interface and three classes.

Program 1 is interface `ArithmeticOperation` containing the two basic arithmetic operations, addition and subtraction, between two numbers. Every implementation class of this interface must be immutable, which allows for certain optimizations.

(1) Type parameter `T` is the type of the argument and return value of the methods.
(2) Method `add` returns the result of addition: the value represented by `this` + the value specified by argument `b`.
(3) Method `sub` returns the result of subtraction: the value represented by `this` – the value specified by argument `b`.

Program 2 is abstract class `ComparableNumber` that partially implements two interfaces. One is `java.lang.Comparable` to be able to sort all instances of this class, and the other is `ArithmeticOperation` to perform arithmetic operations. Class `ComparableNumber` has the following methods.

(1) Abstract method `doubleValue` returns the double value of this `ComparableNumber`. For example, the double value of 1/5 is 0.2.
(2) Method `compareTo` compares two `ComparableNumbers` `a` (`this`) and `b` specified by the arguments. If `a` is greater than `b`, it returns a value greater than 0, if `a` is less than `b`, it returns a value less than 0. Otherwise, it returns 0.

Program 3 is class `DoubleNumber` representing a double (a primitive data type of the Java language). The class extends class `ComparableNumber`.

Program 4 is class `RationalNumber` representing a rational number. The class extends class `ComparableNumber`. A rational number is a number that can be expressed as the quotient p/q of two integers p and q ($q \neq 0$). Class `RationalNumber` contains the following constructor and methods.

(1) Constructor `RationalNumber` creates and initializes a new rational number object with the specified `numerator` and `denominator`. If the specified `denominator` is zero, an `ArithmeticException` is thrown.
(2) Method `greatestCommonDivisor` returns the greatest common divisor of the two specified integers calculated using the Euclidean algorithm.

(3) Method `toString` returns the string representation of this rational number.

(4) Method `add` returns the result of arithmetic operation: `(this + b)`.

(5) Method `sub` returns the result of arithmetic operation: `(this + (-b))`.

[Program 1]
```
public interface ArithmeticOperation<T> {
   public T add(T b);
   public T sub(T b);
}
```

[Program 2]
```
public abstract class ComparableNumber<T> implements [    A    ] {
   public abstract double doubleValue();

   @Override
   public int compareTo(ComparableNumber<?> o) {
      return Double.compare(doubleValue(), o.doubleValue());
   }
}
```

[Program 3]
```
public class DoubleNumber extends ComparableNumber<DoubleNumber> {
   private final double value;

   public DoubleNumber(double value) {
      this.value = value;
   }

   @Override
   public double doubleValue() {
      return value;
   }

   @Override
   public DoubleNumber add(DoubleNumber b) {
      return new DoubleNumber(value + b.value);
   }

   @Override
   public DoubleNumber sub(DoubleNumber b) {
      return new DoubleNumber(value - b.value);
   }
```

```
    @Override
    public String toString() {
        return String.valueOf(value);
    }
}
```

[Program 4]

```
public class RationalNumber extends [    B    ] {
    private final int numerator;
    private final int denominator;

    public RationalNumber(int numerator, int denominator) {
        if (denominator == 0) {
            throw new ArithmeticException("denominator is zero");
        }
        int gcd = greatestCommonDivisor(Math.abs((long)numerator),
                                        Math.abs((long)denominator));
        denominator /= gcd;
        numerator /= gcd;
        if (denominator < 0) {
            denominator = -denominator;
            numerator = -numerator;
        }
        this.numerator = numerator;
        this.denominator = denominator;
    }

    private int greatestCommonDivisor(long a, long b) {
        long m = a;
        long n = b;

        while (n != 0) {
            long temp = [    C    ];
            m = n;
            n = temp;
        }
        return (int) m;
    }
    @Override
    public String toString() {
       StringBuilder sb = new StringBuilder().append(numerator);
       if (denominator != 1) {
          sb.append("/").append(denominator);
       }
       return sb.toString();
    }

    @Override
    public RationalNumber add(RationalNumber b) {
       // optimization for special cases
       if (numerator == 0) {
          return b;
       }
       if (b.numerator == 0) {
          return this;
       }
       // add cross-product terms for numerator
       return new RationalNumber((numerator * b.denominator)
                                 + (b.numerator * denominator),
                                 denominator * b.denominator);
    }

    @Override
    public RationalNumber sub(RationalNumber b) {
       return add(new RationalNumber(    D    ));
    }

    @Override
    public double doubleValue() {
       return (double)    E    ;
    }
 }
```

Subquestion 1

From the answer groups below, select the correct answer to be inserted in each blank $\boxed{}$ in the above programs.

Answer group for A and B

a) `ArithmeticOperation`
b) `ArithmeticOperation, Comparable`
c) `ArithmeticOperation, Comparable<ComparableNumber<?>>`
d) `ArithmeticOperation<T>`
e) `ArithmeticOperation<T>, Comparable`
f) `ArithmeticOperation<T>, Comparable<ComparableNumber<?>>`
g) `Comparable`
h) `Comparable<ComparableNumber<?>>`
i) `ComparableNumber`
j) `ComparableNumber<RationalNumber>`

Answer group for C

a) `(m + n - 1) / m`
b) `(m + n - 1) / n`
c) `m % n`
d) `m / n`
e) `n % m`
f) `n / m`

Answer group for D

a) `-b.denominator, b.numerator`
b) `-b.numerator, b.denominator`
c) `-denominator, numerator`
d) `-numerator, denominator`
e) `b.denominator, b.numerator`
f) `b.numerator, b.denominator`
g) `denominator, numerator`
h) `numerator, denominator`

Answer group for E

a) `(denominator * numerator)`
b) `(numerator / denominator)`
c) `denominator * numerator`
d) `denominator / numerator`
e) `numerator * denominator`
f) `numerator / denominator`

Subquestion 2

From the answer group below, select the correct answer to be inserted in the blank $\boxed{}$ in Program 5.

Program 5 is class `TestNumbers` for testing the above interface and classes. When executing method `main` of this class, it prints out all elements of `numbers` in ascending order.

[Program 5]

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class TestNumbers {
   public static void main(String[] args) {
      List<ComparableNumber<?>> numbers = new ArrayList<>();
      RationalNumber a = new RationalNumber(1, 2);
      RationalNumber b = new RationalNumber(3, 8);
      numbers.add(a);
      numbers.add(b);
      numbers.add(a.add(b));
      numbers.add(a.sub(b));
      DoubleNumber c = new DoubleNumber(0.1);
      numbers.add(c);
      Collections.sort(numbers);
      for (    F    ) {
         System.out.println(number);
      }
      /* α */
   }
}
```

The following lines are produced when method `main` of class `TestNumbers` is executed.

```
0.1
1/8
3/8
1/2
7/8
```

Answer group for F

a) `ComparableNumber<?> number : numbers`
b) `ComparableNumber<DoubleNumber> number : numbers`
c) `ComparableNumber<RationalNumber> number : numbers`
d) `DoubleNumber number : numbers`
e) `int i = 0; i < numbers.size(); i++`
f) `int i = numbers.size() - 1; i >= 0; i--`
g) `RationalNumber number : numbers`

Subquestion 3

Replacing the line `/*` **α** `*/` in Program 5 with the following line will yield an error.

```
System.out.println(a.add(c));
```

From the answer group below, select the correct description of the error caused by replacing the line.

Answer group

a) A `ClassCastException` is thrown at the runtime.
b) A compilation error occurs at the compilation time.
c) A `NullPointerException` is thrown at the runtime.
d) An `ArithmeticException` is thrown at the runtime.
e) An `IllegalArgumentException` is thrown at the runtime.