Q2 2020

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 was the spring Fundamental IT Engineer Examination conducted in 2019?

   Answer group
   a) March b) April c) May d) June

   Since the correct answer is "b) April", 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 in the 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 precedence]

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 zone-based policy firewall (ZPF), and then answer Subquestions 1 through 3.

Over time, networks continued to grow, and they were increasingly used to transfer and store sensitive information. The information and services available are essential to the organization.

A widely accepted host-based security service is the firewall. The most basic type of a firewall uses access control lists (ACLs) to filter IP traffic and monitor established traffic patterns. Later, firewall implementation uses a zone-based approach that operates as a function of interfaces instead of access control lists. In a ZPF, interfaces are assigned to zones, and inspection policy is applied to traffic moving between the zones. A ZPF can take three possible actions that inspect, drop, and pass. It can be configured for extremely advanced, protocol-specific, granular control. It prohibits traffic via a default deny-all policy between different firewall zones. ZPF is suited for multiple interfaces that have similar or varying security requirements.

Subquestion 1

From the answer group below, select the correct answer to be inserted in each blank [ ] in the following description. Here, the answers to be inserted in A1 through A4 should be selected as the correct combination from the answer group for A.

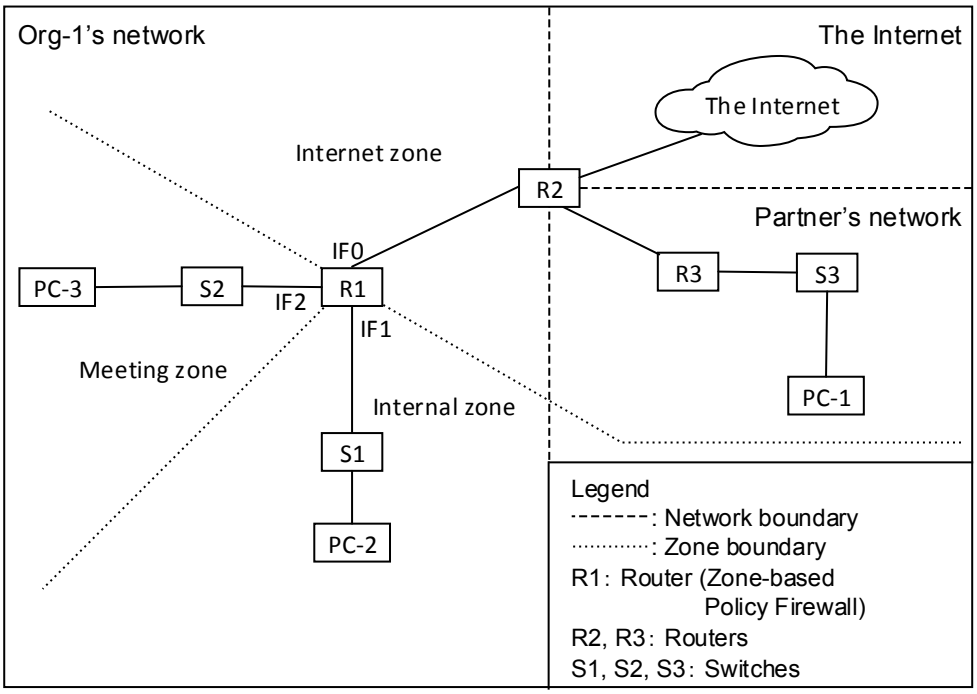


Figure 1 Org-1's network topology

Org-1 is a medium-size organization that is cooperating with a partner organization. Figure 1 shows network devices such as routers, switches and PC hosts in Org-1 and the partner organization. All network devices in Org-1 and the partner organization can communicate with each other, however, they do not provide security. Network configurations enable only end-to-end connectivity.

Therefore, the network security officer for Org-1 decided to implement ZPF to enhance the organization's information security. A ZPF was created on R1, and it is currently responsible for routing packets for the three networks connected to it. R1's interface roles are configured as follows:

- IF0 is connected to the Internet. Because this is a public network, it is considered to be [A1] network and should have [A2] security level.
- IF1 is connected to the internal network. Only authorized users have access to this network. The internal network is considered to be [A3] network and should have [A4] security level.
- IF2 is connected to the meeting network. The meeting network is used only to host meetings with people who are not part of the organization. The meeting network is considered to be untrusted network and should have medium security level.

Answer group for A

	A1	A2	A3	A4
a)	a trusted	the highest	a trusted	the lowest
b)	an untrusted	the highest	an untrusted	the lowest
c)	a trusted	the lowest	a trusted	the highest
d)	an untrusted	the lowest	a trusted	the highest
e)	a trusted	the highest	an untrusted	the lowest
f)	an untrusted	the lowest	an untrusted	the highest

Subquestion 2

From the answer groups below, select the appropriate answer to be inserted in each blank [ ] in the following description. Here, the answers to be inserted in B1 through B3 should be selected as the correct combination from the answer group for B.

The security policy to be enforced by R1 when it is acting as a firewall dictates that:

- No traffic initiated from the [B1] should be allowed into the [B2] or [B3].
- Returning [B1] traffic (return packets coming from the [B1] into the R1 site, in response to requests originating from any of the R1 networks) should be allowed.

- Hosts in the [B2] are allowed to initiate any type of traffic (TCP, UDP or ICMP based traffic).
- Hosts in the [B3] are allowed to initiate only web traffic (HTTP or HTTPS) to the [B1].
- No traffic is allowed between the [B2] and the [B3]. There is no guarantee regarding the condition of guest computers in the [B3]. Such machines could be infected with malware and malicious traffic.

Answer group for B

	B1	B2	B3
a)	internal network	Internet	meeting network
b)	internal network	meeting network	Internet
c)	Internet	internal network	meeting network
d)	Internet	meeting network	internal network
e)	meeting network	internal network	Internet
f)	meeting network	Internet	internal network

Subquestion 3

After ZPF configuration, we need to verify that ZPF functionality is correct. During ZPF verification, we need to know about the self zone. The self zone is the router itself and includes all the IP addresses assigned to the router interfaces. The rules for a ZPF are different for the self zone. If the router is a source or a destination, then all traffic is permitted except when there is a zone-pair with a specific service-policy between the source and destination. In this case, there is no specific service-policy.
From the answer group below, select two correct statements.

Answer group

a) When PC-1 pings PC-2, ICMP packets generated by PC-1's ping are successful.
b) When PC-1 pings PC-3, ICMP packets generated by PC-1's ping are dropped.
c) When PC-1 pings R1's IF1 interface, ICMP packets generated by PC-1's ping are dropped.
d) When PC-2 pings PC-1, ICMP packets generated by PC-2's ping are dropped.
e) When PC-3 pings PC-1, ICMP packets generated by PC-3's ping are dropped.
f) When PC-3 pings R1's IF1 interface, ICMP packets generated by PC-3's ping are dropped.

Q2. Read the following description of resource access controls, and then answer Subquestions 1 and 2.

In a multiprogramming system, there exists a situation where a process halts due to the resource it requests being allocated to other processes and never released by them. This situation is called a deadlock.

The graph called a resource allocation graph is often used to describe the state of the processes and the resources in a system. A resource allocation graph consists of a set of vertices V and a set of edges E. V contains two different subsets of vertices: P = {P1, P2, … , Pn}, consisting of all the running processes in the system, and R = {R1, R2, … , Rm), consisting of all resource types in the system. A directed edge from process P*i* to resource type R*j* is called a request edge. It describes that process P*i* has requested resource type R*j* and is currently waiting for that resource. A directed edge from resource type R*j* to process P*i* is called an assignment edge. It describes that resource type R*j* has been allocated to process P*i*. A system is in a deadlock state if there is a circular path in the graph.

Subquestion 1

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

Figure 1 shows an example of a resource allocation graph. A circle indicates a process and a square indicates a resource type.

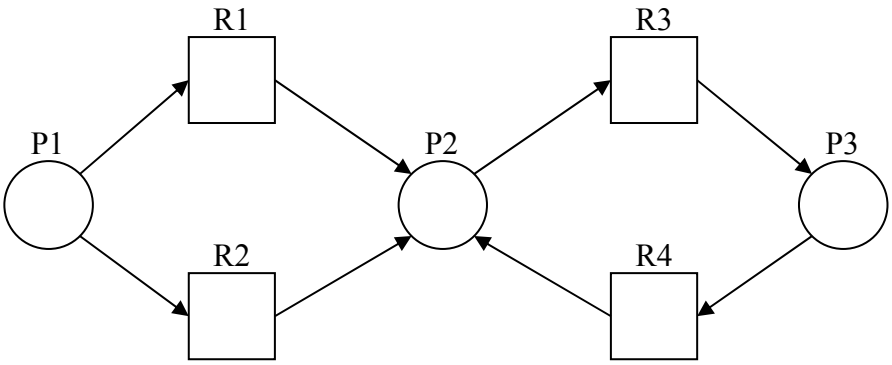


Figure 1 Example of a resource allocation graph

In Figure 1, process P2 is holding [A] and process P3 is waiting for [B]. The path consisting of the vertices [C] makes a circular path, so the system described in Figure 1 is in a deadlock state.

Answer group for A and B

a) R1 and R2
b) R1, R2 and R4
c) R3
d) R4

Answer group for C

a) P1, R1, P2 and R2
b) P1, R1, P2 and R3
c) P1, R2, P2 and R4
d) P2, R3, P3 and R4

Subquestion 2

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

Practically, one resource type can have a set of "instances". For example, personal computers and mobile phones currently ship with a CPU that has two or more processing units and the processes that one unit can execute can also be executed by the others. The resource allocation graph describing such a system has a vertex corresponding to a resource type that has multiple edges directed to multiple processes.

Figure 2 shows an example of a resource allocation graph with multiple-instances resource types.

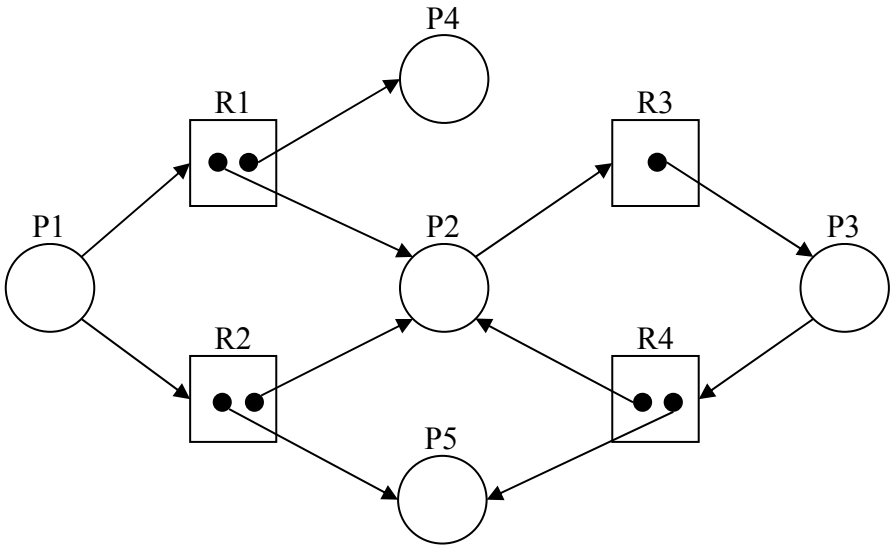


Figure 2 Example of a multiple-instances resource allocation graph

In Figure 2, a vertex corresponding to a resource type is depicted as a square with dots indicating the number of instances. The assignment edge of such a resource type is directed from one of the dots to a process vertex.

The graph has a circular path consisting of [C], however, the system is not necessarily falling into a deadlock state. It is immediately determinable that the system can avoid a deadlock state if process [D] releases the resource instances it currently holds. The system can still fall into a deadlock state, for example, if [E]. Therefore, the circular dependency is not a sufficient condition for deadlocks in a system with a multiple-instance resource allocation graph.

Answer group for D

a) P1 b) P2 c) P3
d) P4 e) P5

Answer group for E

a) a new process P6 is created and starts waiting for resource type R2
b) process P1 cancels all its requests simultaneously
c) process P4 stops holding resource type R1
d) process P5 starts waiting for resource type R3 without releasing the held resources

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

The following is a part of the process for vaccinations in a clinic. At the first visit to the clinic, every customer gets a customer ID in the form YYMMCCCC. YY is the last 2 digits of the year, MM is the month, and CCCC is the serial number of customers in that month. The customer information is stored in the Customer table. The structure of the Customer table and sample data are shown below.

Customer Table

CustomerID	Name	Birthday	Gender	Address
20030495	John Cruise	2019-10-31	M	310 Coconut St.
20040138	Mary Park	2020-01-20	F	41 Orange St.

After registration, a customer meets the clinic's doctor to have a medical examination. Examination results indicate whether the customer is in good enough health for a vaccination or not. Information on examinations is stored in the Examination table. The structure of the Examination table and sample data are shown below.

Examination Table

ExamID	CustomerID	ExamDate	Result	Doctor
11785	20030495	2020-03-25	NG	Gary
12037	20040138	2020-04-07	OK	Tony
12094	20030495	2020-04-10	OK	Gary

Exam ID is a serial number that is generated automatically. Result is either OK (in good health) or NG (not in good health), indicating the health of the customer. If the result is NG, the customer will take the examination again later until the result becomes OK.

Subquestion 1

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

The clinic is planning to check the health status of the customers who took the examination three times or more during the last three months.

The following SQL statement “SQL1” outputs the customer ID, name, and exam count of the customers who took the examination three times or more between January 1, 2020 and March 31, 2020.

```
-- SQL1 --
SELECT Customer.CustomerID, Customer.Name, COUNT(*) AS ExamCount
  FROM Customer, Examination
  WHERE Customer.CustomerID = Examination.CustomerID
    [ A ]
```

Answer group for A

a)
```
AND (ExamDate >= '2020-01-01' OR ExamDate <= '2020-03-31')
GROUP BY Customer.CustomerID, Customer.Name
HAVING COUNT(*) >= 3
```

b)
```
AND ExamDate >= '2020-01-01' AND ExamDate <= '2020-03-31'
AND COUNT(*) >= 3
```

c)
```
AND ExamDate BETWEEN '2020-01-01' AND '2020-03-31'
GROUP BY Customer.CustomerID, Customer.Name
HAVING COUNT(*) >= 3
```

d)
```
GROUP BY Customer.CustomerID, Customer.Name, ExamDate
HAVING ExamDate >= '2020-01-01' AND ExamDate <= '2020-03-31'
       AND COUNT(*) >= 3
```

Subquestion 2

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

The clinic wants to identify the exam status of a customer. The exam status will be either “Ended” or “Follow-up”, depending on whether the result of the latest examination was OK or NG. The following SQL statement “SQL2” outputs the exam status of the customer whose ID is 20030495.

```
-- SQL2 --
SELECT Examination.CustomerID,
       CASE WHEN [ B ] END AS ExamStatus
  FROM Examination
  WHERE Examination.CustomerID = '20030495'
    AND Examination.ExamDate = (SELECT [ C ]
        FROM Examination WHERE Examination.CustomerID = '20030495')
```

Answer group for B

a) `Examination.Result = 'NG' THEN 'Ended' ELSE 'Follow-up'`

b)
```
Examination.Result = 'NG' THEN 'Follow-up'
WHEN Examination.Result != 'NG' THEN 'Ended'
```

c) `Examination.Result != 'NG' THEN 'Ended' OTHERWISE 'Follow-up'`

d)
```
Examination.Result != 'NG' THEN 'Follow-up'
WHEN Examination.Result = 'NG' THEN 'Ended'
```

Answer group for C

a) `DISTINCT Examination.ExamDate` b) `Examination.ExamDate`

c) `MAX(Examination.ExamDate)` d) `MIN(Examination.ExamDate)`

Subquestion 3

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

The clinic is developing the following SQL statement "SQL3" that will output the number of customers who are babies under 90 days old at the present time and took the examination(s).
Here, the function `DATEDIFF(day, date1, date2)` returns the difference, in days, between *date1* and *date2*, and the function `GETDATE()` returns the current date.

Line No.

```
1   -- SQL3 --
2   SELECT COUNT(Customer.CustomerID)
3     FROM Customer
4     JOIN Examination ON Customer.CustomerID = Examination.CustomerID
5     WHERE DATEDIFF(day, GETDATE(), Customer.Birthday) < 90
```

When SQL3 is executed, an incorrect number of customers is displayed. It is found that a customer is counted multiple times if the customer took examinations multiple times. This problem can be resolved by [D]. Here, the `LIMIT` clause limits the number of rows to return.

Answer group for D

a) adding `LIMIT 1` after line 5

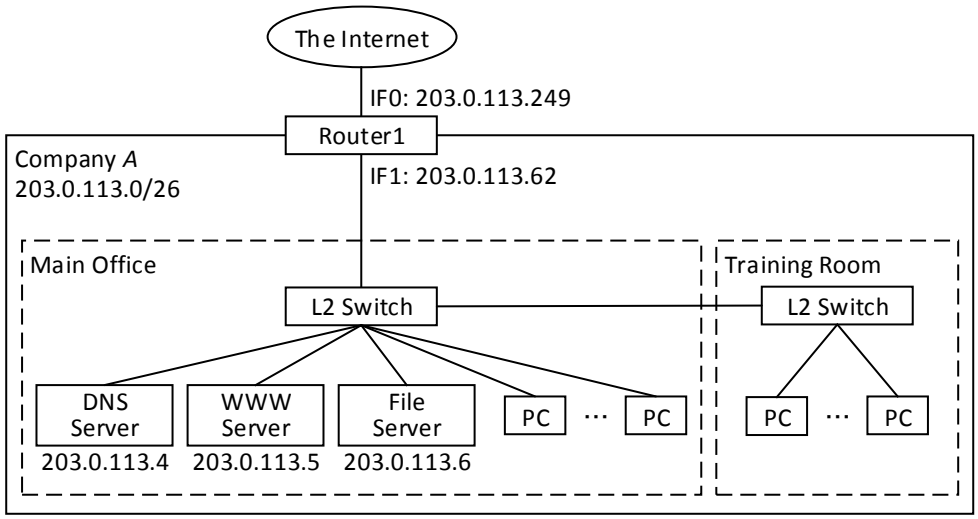
b) changing `COUNT(Customer.CustomerID)` to `COUNT(*)` on line2

c) changing `COUNT(Customer.CustomerID)` to `COUNT(*)` on line2
and adding `LIMIT 1` after line 5

d) changing `COUNT(Customer.CustomerID)`
to `COUNT(DISTINCT Customer.CustomerID)` on line 2

Q4. Read the following description of the network configuration of company *A*, and then answer Subquestions 1 through 3.

Company *A* is a training solutions provider currently setting up a computer network in its new location. Recently, company *A* allocated a group of public IP addresses with the network address 203.0.113.0/26. Initially, company *A* used the same network address space for both its main office and the training room since only a few PCs were required at the beginning of its operation. The initial network configuration of company *A* is shown in Figure 1.

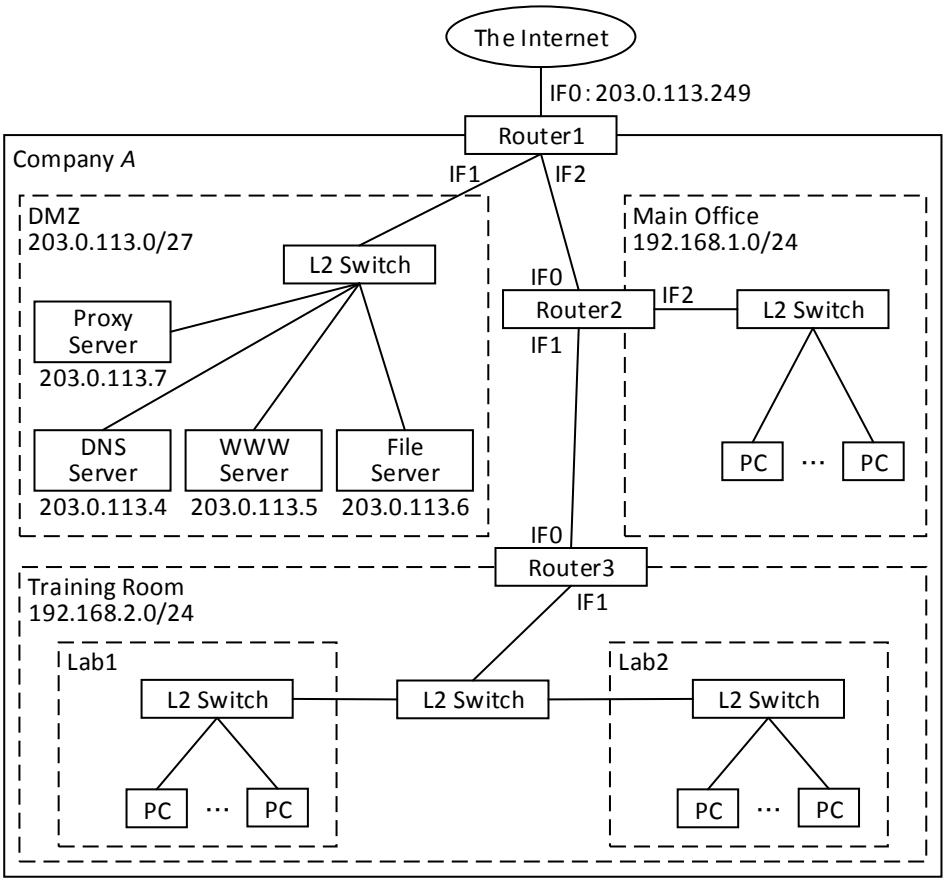


Note: Some information is intentionally omitted.

Figure 1 Initial network configuration of Company *A*

The IP address 203.0.113.249 is assigned to the WAN interface, IF0, of the router by the ISP and 203.0.113.62 is assigned to the LAN interface, IF1, of the router. With all the occupied IP addresses including the three servers with specific IP addresses, as shown in Figure 1, there are [A] available public IP addresses that can be assigned to the PCs in company *A*.

Afterward, the management of company *A* decided to offer more training courses, which in turn needed to have more PCs than the available public IP addresses. The management also decided to separate all the publicly accessible servers into a DMZ segment. The revised network configuration of company *A* is shown in Figure 2.



Note: Some information is intentionally omitted.

Figure 2 Revised network configuration of Company *A*

In the revised configuration, the subnet 203.0.113.0/27 is assigned to the DMZ. Here, assume that all the firewall rules on the routers are set appropriately to allow incoming connections to the DMZ only, not to other segments. The rest of the public address space is reserved for future use. Here, the last usable IP address of the DMZ is assigned to IF1 of Router1. Since the IP address assigned to IF1 of Router1 is now 203.0.113. B , it is necessary to update the gateway address on all servers in the DMZ to reflect this change as well.

Since the PCs in both the main office and training rooms are not required to be directly accessible from the Internet, the private address spaces 192.168.1.0/24 and 192.168.2.0/24 are assigned to them respectively. The proxy server is deployed in the DMZ segment so that PCs can access the Internet. Table 1 shows the final stage of network configurations on some of the servers in the DMZ and a PC in the main office.

Table 1 Network configurations on some of the servers in the DMZ and a PC in the main office

	WWW Server	File Server	Proxy Server	PC in the Main Office
IP Address	203.0.113.5	203.0.113.6	203.0.113.7	192.168.1.8
Subnet Mask	255.255.255. C	255.255.255. C	255.255.255. C	255.255.255.0
DNS Server	203.0.113.4	203.0.113.4	203.0.113.4	203.0.113.4
Gateway	203.0.113. B	203.0.113. B	203.0.113. B	192.168.1.1

Subquestion 1

From the answer group below, select the correct answer to be inserted in each blank [ ] in the above description and Table 1.

Answer group for A through C

a) 14 b) 30 c) 31
d) 46 e) 58 f) 62
g) 192 h) 224 i) 249

Subquestion 2

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

Each entry of routing tables has two information fields:

(1) Destination: the network address of the destination. Note that 0.0.0.0/0 for the destination means that the next hop is the default gateway.
(2) Next hop: there are two types of next hop: “Connected” or an IP address.
“Connected” indicates that the destination network is directly connected to the router. On the other hand, an IP address indicates that another router on the directly connected network has the IP address and the router is one hop closer to the destination network.

Table 2 shows some of the IP addresses assigned to the interfaces on the routers. Table 3 shows some of the entries of the routing table on each router.

Table 2 IP addresses assigned to the interfaces on the routers

Router	Interface	IP address
Router1	IF0	203.0.113.249
Router1	IF2	192.168.0.5
Router2	IF0	192.168.0.6
Router2	IF1	192.168.0.9
Router2	IF2	192.168.1.1
Router3	IF0	192.168.0.10
Router3	IF1	192.168.2.1

Table 3 Some of the entries of the routing table on each router

Router	Destination	Next Hop
Router1	192.168.0.0/29	Connected
Router1	192.168.0.8/29	D
Router1	192.168.1.0/24	D
Router1	192.168.2.0/24	D
Router1	203.113.0.0/27	Connected
Router2	0.0.0.0/0	192.168.0.5
Router2	192.168.0.0/29	Connected
Router2	192.168.0.8/29	Connected
Router2	192.168.1.0/24	Connected
Router2	192.168.2.0/24	192.168.0.10
Router3	0.0.0.0/0	E
Router3	192.168.0.8/29	Connected
Router3	192.168.2.0/24	Connected

Note: Some information is intentionally omitted.

Answer group for D and E

a) 192.168.0.5
b) 192.168.0.6
c) 192.168.0.9
d) 192.168.0.10
e) 192.168.1.1
f) 192.168.1.2
g) 203.0.113.249
h) Connected

Subquestion 3

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

The management also decided to move the file server from the DMZ to the main office segment to prevent access from outside because the file server keeps documents that are only for internal use. The IP address 192.168.1.6 is assigned to the file server. Table 4 shows the network configuration on the file server after the relocation.

Table 4 Network configuration on the file server

	File Server
IP Address	192.168.1.6
Subnet Mask	255.255.255.0
DNS Server	203.0.113.4
Gateway	192.168.1.254

All PCs in the main office can now access the file server. However, the file server is unable to access the proxy server and the PCs from the training rooms are unable to access the file server. The administrator confirmed that the firewall rules on the router and the DNS record for the file server on the DNS server were updated properly. Later on, the administrator changed [F] to resolve the problem, so everything performs correctly as a result.

Answer group for F

a) the DNS server of the file server to 192.168.1.1
b) the gateway address of the file server to 192.168.1.1
c) the gateway address of the file server to the same value as the WWW server
d) the IP address of the file server to 192.168.2.6
e) the IP address of the file server to 203.0.113.36
f) the subnet mask of the file server to the same value as the WWW server

Q5. Read the following description of an online student system, and then answer Subquestions 1 and 2.

University R plans to develop an online student system for the registrar office, which serves the students' needs on registration and withdrawal.

[Main Functional Description]

(1) Login
- A student has to login to the system with student ID and password.

(2) Register Course
- The student can register courses by selecting courses using course ID.
- The student can register multiple courses at the same time. The total number of credits must not be greater than the maximum credits, 24 per semester.
- The system generates the course list that the student is able to select. The course list shows:
  - The prerequisite courses that are not cancellable
  - Non-prerequisite courses that have available seats
- JavaScript is used for processing the course list. When the student selects or cancels any courses, it checks the time schedule and the total number of registered credits for this particular semester.
- The system checks whether the added courses satisfy the following conditions:
  - Prerequisite courses
  - Time schedule
  - Number of available seats
  - Total number of registered credits for this particular semester.

  If there are any errors, the system displays messages indicating the courses that were not allowed to be added or needed to be re-selected.
- The student submits the selected course information to the system.
- The system checks the course information. If there are errors, it generates error messages. If there is no error, it updates the database.
- The system shows a confirmation message of the course registration, and sends the course information to the student by email.
- The system allows the student to use this registration function only during the first two weeks of each semester.

(3) Withdraw Course

- During the third week of each semester, the student is allowed to withdraw any courses from the registered courses. To do so, the student has to select the course(s) that he/she wants to withdraw.
- The withdrawals have to be approved by the lecturer.

[Activity Diagram]

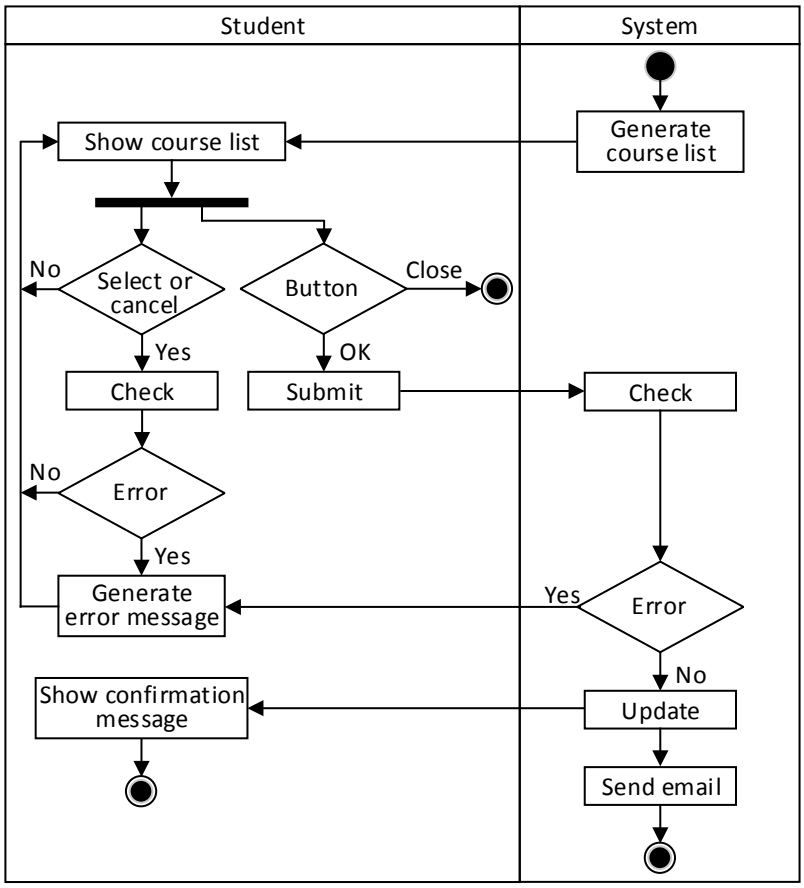


Figure 1 Activity diagram of "Register Course"

Table 1 Use case description of "Register Course"

Diagram name	Register Course
Input	student ID, course ID
Output	The course information
Trigger	The student selects the "Register Course" menu button on the screen.
Availability	First two weeks of semester
Pre-condition	Successful login
Post-condition	Registration and notification of the course information.
Basic flow	Refer to Table 2

Table 2 Description of basic flow of “Register Course”

Student	System
	1. Generate the course list by student ID. Select [A] courses, and the other courses that satisfy [B]. 2. Display the course list.
3. Select a course or cancel course selection. Each time a course is selected or canceled, [C] and [D] are checked. Then, the system checks various conditions. 4. Touch the “OK” button to submit student ID and selected course IDs, or touch the “Close” button to discard the course selection.	
	5. Check the course information. If there is no error, [E].
6. When the courses are registered successfully, then show the success message and logout. Otherwise, check the error messages and proceed to 2.	
	7. After all the registered courses are approved, show the message and send the course information.

Subquestion 1

From the answer groups below, select the correct answer to be inserted in each blank [ ] in Table 2.

Answer group for A through D

a) non-prerequisite
b) number of maximum credits
c) prerequisite
d) seat availability
e) time schedule
f) total number of registered credits
g) week of the semester

Answer group for E

a) display the course list
b) send the course information
c) show the confirmation message
d) update the database

Subquestion 2

From the answer group below, select the information to be received from the system when checking the input items in the course list.

Answer group

a) number of available seats
b) start date of the semester
c) time schedule

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

In a multiprogramming environment, when resources requested by a process cannot be satisfied, the process is put in the waiting state. A deadlock occurs when resources requested by a process are held by other processes that are in the waiting state.
The program `Banker` checks the given status of processes and resources, and displays the execution sequence of the processes if they are not in a deadlock state.

[Program description]

(1) The program `Banker` uses the following arrays. Indexes of all arrays start at 0.

(i) `maxP[][]` and `allocP[][]`:

The resource allocation status of each process is given by these arrays.
`maxP[`*p*`][`*r*`]` = *n* indicates that process P*p* requires *n* instances of resource R*r* to finish the process. `allocP[`*p*`][`*r*`]` = *n* indicates that currently *n* instances of resource R*r* are allocated to process P*p*.
Figure 1a shows an example of process status. There are 3 processes (P0, P1, and P2) and 3 types of resources (R0, R1, and R2). For example, `maxP` shows that process P0 requires four R0, one R1, and one R2 resources to finish the process, and `allocP` shows that two R0, one R1, and no R2 resources are currently allocated to process P0. Therefore, P0 is waiting for two more R0 and one more R2 resources.

maxP[][]:	R0	R1	R2
P0	4	1	1
P1	1	3	1
P2	1	1	2

Figure 1a Process status

allocP[][]:	R0	R1	R2
P0	2	1	0
P1	1	1	1
P2	0	1	2

	R0	R1	R2
maxR[]:	5	4	3
allocR[]:	3	3	3
availR[]:	2	1	0

Figure 1b Resource status

(ii) `maxR[]`, `allocR[]` and `availR[]`:

The current allocation status of each resource is given by these three arrays.
`maxR[`*r*`]` = *n* indicates that there are maximum *n* instances of resource R*r*.
`allocR[`*r*`]` = *n* indicates that *n* instances of resource R*r* are allocated to processes.
`availR[`*r*`]` = *n* indicates that *n* instances of resource R*r* are currently available.
Figure 1b, that is connected with Figure 1a, shows an example of resource status. As for resource R0, among a maximum of five R0 resources, three of them are allocated to the processes (that is, as in Figure 1a, two for P0 and one for P1), therefore, two (=5 – 3) R0 resources are currently available for allocation.

(iii) `stateP[]`:

`stateP[`*p*`]` = 1 indicates that process P*p* is waiting for execution.

`stateP[`*p*`]` = 0 indicates that process P*p* is executed and finished.

Initially, all elements of `stateP` are set to 1.

(2) Let `pn` be the number of processes, and `rn` be the number of resources. The values in arrays `maxP`, `allocP`, and `maxR` are set in advance. First, set the values in arrays `allocR` and `availR` from the values in arrays `maxP`, `allocP`, and `maxR`.

(3) Then, repeat this step while the number of value 1s in `stateP` > 0. If the number of value 1s in `stateP` reaches 0, go to (4).

(i) Check arrays `maxP` and `allocP` from top to bottom (excluding the finished processes that have value 0 in `stateP`) to find the next process that is executable. Process `p` is determined as executable if the following allocation condition is met:

$$\boxed{\text{A}} - \boxed{\text{B}} \leq \texttt{availR[r]} \quad (\texttt{r}: 0, 1, \ldots, \texttt{rn}-1)$$

If the next executable process exists, proceed to (ii), otherwise, go to (4).

(ii) For process `p` that is determined as executable, perform the following settings:

Free the resources held by process `p` and update the contents of `availR` as follows:

$$\texttt{availR[r]} \leftarrow \texttt{availR[r]} + \boxed{\text{C}} \quad (\texttt{r}: 0, 1, \ldots, \texttt{rn}-1)$$

Set `StateP[p]` to 0.

Display the message "`Process Pp is executed and finished.`".

(4) Terminate the program. In case if processes that are waiting for execution still remain, display "`The processes are in deadlock state.`" before termination.

[Program]

```
PROGRAM: Banker() {
   INT: p, pn ← 3  /* pn: number of processes */
   INT: r, rn ← 3  /* rn: number of resources */
   INT:   maxP[pn][rn] ← {{4, 1, 1}, {1, 3, 1}, {1, 1, 2}}
   INT: allocP[pn][rn] ← {{2, 1, 0}, {1, 1, 1}, {0, 1, 2}}
   INT: stateP[pn] ← {1, 1, 1}  /* set "waiting for execution" state */
   INT:   maxR[rn] ← {5, 4, 3}  /* max number of each resource */
   INT: allocR[rn] ← {0, 0, 0}
   INT: availR[rn]
   INT: number_of_value_1s_in_stateP
   BOOLEAN: deadlock_occurs, executable_process_is_found
```

```
    FOR (p ← 0; p < pn; p ← p + 1) {  /* set allocatable resources */
       FOR (r ← 0; r < rn; r ← r + 1) {
          allocR[r] ← allocR[r] + allocP[p][r];
       }
    }
    FOR (r ← 0; r < rn; r ← r + 1) {  /* set available resources */
       availR[r] ← maxR[r] - allocR[r];
    }

    number_of_value_1s_in_stateP ← pn;

    print(maxP[][], allocP[][]);  /* print the contents of arrays in edited form */
    print(maxR[], allocR[], availR[], stateP[]);

    WHILE (number_of_value_1s_in_stateP > 0) {
       deadlock_occurs ← true;
       FOR (p ← 0; p < pn; p ← p + 1) {
          IF (stateP[p] = 1) {
             executable_process_is_found ← true;
             FOR (r ← 0; r < rn; r ← r + 1) {
                IF ( [ A ] - [ B ] > availR[r]) {
                   executable_process_is_found ← false;
                   break;  /* exit the inner FOR loop */
                }
             }
             IF (executable_process_is_found) {
                deadlock_occurs ← false;
                FOR (r ← 0; r < rn; r ← r + 1) {
                   availR[r] ← availR[r] + [ C ];
                }
                [ D ];
                number_of_value_1s_in_stateP ←
                   number_of_value_1s_in_stateP - 1;
                print("Process P", p, " is executed and finished.");
                break;  /* exit the FOR loop */
             }
          }
       }

       IF (deadlock_occurs) {
          print("The processes are in deadlock state.");
          break;  /* exit the WHILE loop */
       }
       print(availR[], stateP[]);  /* print the contents of arrays */
    }
}
```

Figure 2 shows the program output for the case shown in Figures 1a and 1b.

```
  maxP: R0 R1 R2  allocP: R0 R1 R2
    P0   4  1  1      P0   2  1  0
    P1   1  3  1      P1   1  1  1
    P2   1  1  2      P2   0  1  2

        R0 R1 R2
  maxR:  5  4  3
allocR:  3  3  3          P0 P1 P2
availR:  2  1  0  stateP:  1  1  1

Process P2 is executed and finished.

        R0 R1 R2          P0 P1 P2
availR:  2  2  2  stateP:  1  1  0

Process [   E   ] is executed and finished.

        R0 R1 R2          P0 P1 P2
availR: [   F   ] stateP: [       ]

Process [       ] is executed and finished.

        R0 R1 R2          P0 P1 P2
availR:  5  4  3  stateP:  0  0  0
```

Note: Shaded parts are not shown

Figure 2 Program output for the data shown in Figures 1a and 1b

Subquestion 1

From the answer groups below, select the correct answer to be inserted in each blank [ ] in Figure 2 and the program. If necessary, select the same answer twice or more.

Answer group for A, B and C

a) `allocP[p][r]`
b) `allocR[r]`
c) `maxP[p][r]`
d) `maxR[r]`

Answer group for D

a) `executable_process_is_found ← false`
b) `stateP[p] ← 0`
c) `stateP[p] ← 1`

Answer group for E

a) P0 b) P1 c) P2

Answer group for F

a) 2 3 2 b) 2 4 3 c) 3 3 3

d) 3 4 2 e) 4 3 2 f) 4 4 3

Subquestion 2

From the answer groups below, select the correct answer to be inserted in the blank [ ] in the following description.

When the program Banker is executed by setting the values shown in Figures 3a and 3b to arrays maxP, allocP, and maxR, the execution result reveals that [G].

maxP[][]:	R0	R1	R2
P0	4	1	1
P1	1	3	1
P2	1	1	2

Figure 3a Process status

allocP[][]:	R0	R1	R2
P0	2	1	0
P1	1	1	0
P2	1	0	1

	R0	R1	R2
maxR[]:	4	3	2
allocR[]:	4	2	1
availR[]:	0	1	1

Figure 3b Resource status

Answer group for G

a) P0, P1, and P2 are executed and finished

b) P0, P1, and P2 are in a deadlock state

c) P0 is executed and finished; however, P1 and P2 are in a deadlock state

d) P1 is executed and finished; however, P0 and P2 are in a deadlock state

e) P2 is executed and finished; however, P0 and P1 are in a deadlock state

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 Subquestion.

The shortest path problem is applied to find an efficient routing in various network. For example, car navigation systems or network routers need to solve the problem to obtain the optimal routing for cars or packets. To formulate a network, a graph is often used. A graph is a set of nodes and edges. Dijkstra's algorithm is a well-known method to obtain the shortest distance from one node, the starting point, to the other nodes in the graph.

[Program Description]

A sample network is shown in Figure 1. The network is composed of 6 nodes and 10 edges. A number in a circle indicates a node number, and a number along an edge indicates the edge length. When a network has `N` nodes, nodes are numbered 0, 1, …, `N-1`.

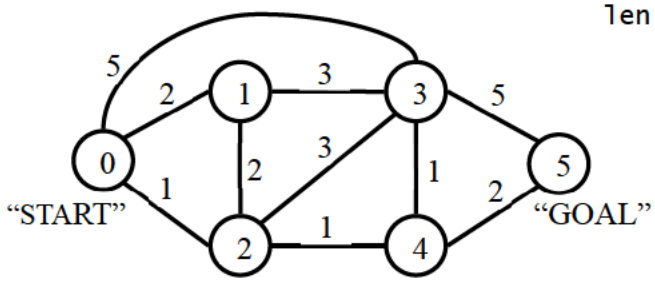


Figure 1 Sample network

`len[][]`:

	0	1	2	3	4	5
0	0	2	1	5	INF	INF
1	2	0	2	3	INF	INF
2	1	2	0	3	1	INF
3	5	3	3	0	1	5
4	INF	INF	1	1	0	2
5	INF	INF	INF	5	2	0

Figure 2 Internal expression of the sample network

The program description shown below refers to the sample network in Figure 1 as an example. Information about the sample network in Figure 1 is stored in the array `len[][]` in advance as shown in Figure 2. If nodes *i* and *j* are connected directly, `len[`*i*`][`*j*`]` stores the edge length between nodes *i* and *j*. If nodes *i* and *j* are not connected directly, `len[`*i*`][`*j*`]` stores the maximum integer value INF.

To obtain the shortest distances from node 0 (the starting point) to other nodes, the program uses the arrays `visited[]` and `dist[]`.

`visited[]` stores 0 or 1; 0 indicates that the node has not yet been visited, and 1 indicates that the node has been visited and processed. Initially, all nodes are set to 0, as shown in Figure 3. `dist[]` stores the distance from node 0 to each node at the present time. Initially, `dist[0]` is set to 0, and every other `dist[`*i*`]` (*i* > 0) is set to INF, as shown in Figure 3.

After setting the initial values in `visited[]` and `dist[]`, the program repeats the following steps (1) and (2) `N` times. Figure 3 shows how the steps proceed.

(1) Among the nodes that have value 0 in `visited[]`, select a node whose value in `dist[]` is the shortest. Let the selected node number be *i*, and set 1 to `visited[`*i*`]`.

(2) For node *i* selected in (1), check every node *j* that is directly connected to node *i*, and if the sum of the distances from node 0 to node *i* and from node *i* to node *j* is less than the current distance from node 0 to node *j*, then replace the current distance from node 0 to node *j* with the shorter distance via node *i*.

Checkpoint	`visited[]`						`dist[]`					
	0	1	2	3	4	5	0	1	2	3	4	5
Initial settings	0	0	0	0	0	0	0	INF	INF	INF	INF	INF
After 1st loop	1	0	0	0	0	0	0	2	1	5	INF	INF
After 2nd loop	1	0	1	0	0	0	0	2	1	4	2	INF
After 3rd loop	1	1	1	0	0	0	0	2	1	4	2	INF
After 4th loop	1	1	1	A1	0		0	2	1	3	2	A2
After 5th loop	1	1	1	1	1	0	0	2	1	3	2	4
After 6th loop	1	1	1	1	1	1	0	2	1	3	2	4

Figure 3 Steps to obtain the shortest distances from the starting point

Finally, the shortest distances from node 0 to other nodes are obtained in `dist[]`, and the program displays the following message. Here, node 5 is the GOAL node.

```
The shortest distance from START to GOAL is 4.
```

The program has the following functions:

(1) `void dijkstra(int len[][N], int start, int dist[]);`
Solves the shortest path problem using Dijkstra's algorithm. It calls the functions `choose` and `update`. The node number of the starting point is given by `start`.

(2) `int choose(int visited[], int dist[]);`
Selects one node which has the shortest distance among the nodes that have not yet been visited, updates the contents of the array `visited[]`, and returns the selected node number.

(3) `void update(int i, int visited[], int len[][n], int dist[]);`
Updates the distance to the nodes directly connected to the selected node number given by `i`.

[Program]

```
#include <stdio.h>
#include <limits.h>

#define N   6
#define INF INT_MAX

void dijkstra(int len[][N], int start, int dist[]);
int choose(int visited[], int dist[]);
void update(int i, int visited[], int len[][N], int dist[]);

int main() {
   int dist[N];
   /* corresponds to the network in Figure 1 */
   int len[N][N] = {{   0,   2,   1,   5, INF, INF },
                    {   2,   0,   2,   3, INF, INF },
                    {   1,   2,   0,   3,   1, INF },
                    {   5,   3,   3,   0,   1,   5 },
                    { INF, INF,   1,   1,   0,   2 },
                    { INF, INF, INF,   5,   2,   0 }};

   dijkstra(len, 0, dist);
   printf("The shortest distance from START to GOAL is %d.", dist[5]);
   return 0;
}

void dijkstra(int len[][N], int start, int dist[]) {
   int i, visited[N], nvisited;

   /* initialization */
   for (i = 0; i < N; i++) {
      visited[i] = [      B      ];
      dist[i] = INF;
   }
   /* start has 0 distance from itself */
   dist[start] = 0;
   /* visit each node */
   for ([      C      ]) {
      i = choose(visited, dist);
      update(i, visited, len, dist);
   }
}
```

```
int choose(int visited[], int dist[]) {
   int i, j, min_dist = INF;

   for (j = 0; j < N; j++) {
      if (visited[j] == 0 && dist[j] < min_dist) {
         min_dist = dist[j];
         [ D ];
      }
   }
   visited[i] = 1;
   return i;
}

void update(int i, int visited[], int len[][N], int dist[]) {
   int j, new_dist_j;
   for (j = 0; j < N; j++) {
      if ( [ E ] ) {
         new_dist_j = [ F ];
         if (visited[j] == 0 && new_dist_j < dist[j]) {
            dist[j] = new_dist_j;
         }
      }
   }
}
```

Subquestion

From the answer groups below, select the correct answer to be inserted in each blank [ ] in Figure 3 and the program. Here, the answers to be inserted in A1 and A2 should be selected as the appropriate combination from the answer group for A.

Answer group for A

	A1	A2
a)	0 1	4
b)	0 1	INF
c)	1 0	4
d)	1 0	INF

Answer group for B

a) `-1`
b) `0`
c) `1`
d) `INF`

Answer group for C

a) `i = 0; i < N; i++`
b) `i = start; i < N; i++`
c) `nvisited = 0; nvisited < N; nvisited++`
d) `nvisited = start; nvisited < N; nvisited++`

Answer group for D

a) `break`
b) `dist[j] = dist[i]`
c) `dist[j]++`
d) `i = j`
e) `return i`
f) `return j`

Answer group for E

a) `i != j`
b) `i == j`
c) `len[i][j] != INF`
d) `len[i][j] == 0`
e) `len[i][j] == INF`
f) `len[i][j] == len[j][i]`

Answer group for F

a) `dist[i] + len[i][j]`
b) `dist[i] - len[i][j]`
c) `dist[j] + len[i][j]`
d) `dist[j] – len[i][j]`

Q8. Read the following description of Java programs and the programs themselves, and then answer Subquestions 1 and 2.

[Program Description]

This program, consisting of two classes, iteratively generates all solutions of the 4-Queen Puzzle, which is the challenge of placing 4 queens on a 4×4 chessboard so that any of the queens cannot attack the other queens. A queen can move in a straight line, horizontally, vertically, or diagonally. Therefore, multiple queens cannot be placed on the same row, the same column, or diagonal positions on a chessboard.

The upper-left corner of the 4×4 chessboard has the coordinates, row = 0, column = 0. The lower-right corner of the 4×4 chessboard has the coordinates, row = 3, column = 3. Coordinates are printed in (row, column) format in the output. The upper-left and lower-right corners should be printed as (0, 0) and (3, 3), respectively.

`Program 1` is the `Play` class that facilitates the puzzle to play.

`Program 2` is the `ChessBoard` class that represents a chessboard for the 4-Queen Puzzle having the following members:

(1) `solutionQuantity`: a field to keep a count of generated solutions.
(2) `BOARD_SIZE`: a constant with the value 4, denoting a 4×4 chessboard.
(3) `queens`: an array of `Queen`s to keep track of the four queens.
(4) `Queen`: a nested class representing a single queen with the following members:
  (i) `row`: a field indicating the row position of this `Queen`
  (ii) `column`: a field indicating the column position of this `Queen`
  (iii) The constructor creates and places a `Queen` on column 0 on the specified row.
  (iv) The getter and setter methods for `column`
  (v) The `canAttack` method returns true if this `Queen` *can* attack the specified `Queen`, or false otherwise.
  (vi) The `toString` method returns the `String` representation of this `Queen`.
(5) The constructor creates and places 4 queens on column 0 of each row.
(6) The `toString` method returns the `String` representation of the coordinates of all the queens on the chessboard.
(7) The `isValidPlacement` method returns true if any of the 4 queens *cannot* attack the other ones, or false otherwise.

(8) The `generateSolutions` method calls the `tryColumnsOf` method that repeatedly places each queen on the first column of different rows, and then moves them to the right to find out all possible valid placements of the queens.

(9) The `printSolution` method prints the current positions of the queens as a valid solution.

(10) The `getRangeStream` method generates a `Stream<Integer>` of the specified `int` range.

Executing the `main` method of the `Play` class produces the following output.

```
Solution No. 1: (0, 1), (1, 3), (2, 0), (3, 2)
EQEE
EEEQ
QEEE
EEQE

Solution No. 2: (0, 2), (1, 0), (2, 3), (3, 1)
EEQE
QEEE
EEEQ
EQEE
```

[Program 1]

```
public class Play {
   public static void main(String[] args) {
      ChessBoard cb = new ChessBoard();
      cb.generateSolutions();
   }
}
```

[Program 2]

```
import java.util.Arrays;
import java.util.stream.Collectors;
import java.util.stream.IntStream;
import java.util.stream.Stream;
```

```
class ChessBoard {
   private static final int BOARD_SIZE = 4;
   private static final String DELIMITER = ", ";
   private static final char QUEEN = 'Q';
   private static final char EMPTY = 'E';

   private int solutionQuantity;
   private Queen[] queens;

   static class Queen {
      private final int row;
      private int column;

      Queen(int row) { this.row = row; }

      private void setColumn(int column) { this.column = column; }

      int getColumn() { return column; }

      boolean canAttack(Queen other) {
         return column == other.column || [ A ];
      }

      @Override
      public String toString() {
         return "(" + row + DELIMITER + column + ")";
      }
   }

   ChessBoard() {
      // Populates queens[] with per row Queen instances
      queens = getRangeStream()
             // map creates a Queen with each row index
             // and returns a Stream<Queen>
             .map( [ B ] )
             // toArray collects the Queen instances into a Queen[]
             .toArray(Queen[]::new);
   }
```

```
@Override
public String toString() {
   String s;
   s = Stream.of(queens)
             // Converts a Queen to its String representation
             .map(Queen::toString)
             // Concatenates all Strings using DELIMITER
             // followed by a line separator (e.g. "a" "b" to "a, b\n")
             .collect(Collectors.joining(DELIMITER,
                         "", System.lineSeparator()));
   char[] boardRow = new char[BOARD_SIZE];
   StringBuilder sb = new StringBuilder(s);
   getRangeStream().forEach(i -> {
      Arrays.fill(boardRow, EMPTY);
      boardRow[    C    ] = QUEEN;
      sb.append(boardRow).append(System.lineSeparator());
   });
   return sb.toString();
}

private boolean isValidPlacement() {
   return getRangeStream(0, BOARD_SIZE - 1)
      // noneMatch returns true if none of the given conditions is true.
      .noneMatch(i -> getRangeStream(i + 1, BOARD_SIZE)
          // anyMatch immediately returns true if any given condition is found true.
          .anyMatch(j -> queens[i].canAttack(queens[    D    ])));
}

private void printSolution() {
   System.out.printf("Solution No. %d: %s%n",
         ++solutionQuantity,    E    );
}

void generateSolutions() {
   tryColumnsOf(0);
}
```

```
    private void tryColumnsOf(int row) {
       getRangeStream().forEach(col -> {
          queens[row].setColumn(col);
          if (row + 1 < BOARD_SIZE) {
             tryColumnsOf(row + 1);
          } else if (isValidPlacement()) {
             printSolution();
          }
       });
    }

    private Stream<Integer> getRangeStream() {
       return getRangeStream(0, BOARD_SIZE);
    }

    private Stream<Integer> getRangeStream(int begin, int end) {
       // Generates a Stream<Integer> of the specified int range
       // from begin (inclusive) to end (exclusive)
       return IntStream.range(begin, end).boxed();
    }
}
```

Subquestion 1

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

Answer group for A

a) `Math.abs(row - other.row) == Math.abs(column - other.column)`

b) `row - column == other.row - other.column`

c) `row + column == other.row + other.column`

d) `row == other.row`

e) `row - other.row == column - other.column`

Answer group for B

a) `i -> Queen(i)`

b) `new Queen()`

c) `Queen::new`

d) `queens[i]::new`

e) `queens[i] = new Queen(i)`

Answer group for C

a) `i`
b) `queens[i]`
c) `queens[i].getColumn()`
d) `queens[i].row`

Answer group for D

a) `i`
b) `i - 1`
c) `i + 1`
d) `j`
e) `j - 1`
f) `j + 1`

Answer group for E

a) `Arrays.asList(queens)`
b) `ChessBoard`
c) `queens`
d) `String.of(queens)`
e) `this`

Subquestion 2

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

In the `ChessBoard` class, the `tryColumnsOf` method is recursively called. The maximum depth of the recursive call is [F], where the first call from the `generateSolutions` method is counted as depth one, and the `isValidPlacement` method is called from the `tryColumnsOf` method [G] times.

Answer group for F and G

a) 2
b) 4
c) 8
d) 16
e) 32
f) 64
g) 128
h) 256