October 2011

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 autumn 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
○		Declares names, types, etc. of procedures, variables, etc.
`/* text */`		Describes comments in text.
Process	• variable ← expression	Assigns the value of an expression to a variable.
	• procedure(argument, ...)	Calls the procedure and passes/receives argument.
	▲ conditional expression ↕ process ▼	Indicates a one-way selection process. If the conditional expression is `true`, then the process is executed.
	▲ conditional expression ↕ process 1 process 2 ▼	Indicates a two-way selection process. If the conditional expression is `true`, then process 1 is executed. If it is `false`, then process 2 is executed.
	■ conditional expression ↕ process ■	Indicates a pre-test iteration process. While the conditional expression is `true`, the process is executed repeatedly.
	■ ↕ process ■ conditional expression	Indicates a post-test iteration process. The process is executed, and then while the conditional expression is `true`, the process is executed repeatedly.
	■ variable: init, cond, incr ↕ process ■	Indicates an iteration process. The initial value init (given by an expression) is stored in the variable at the start of the processing, and then while the conditional expression cond is `true`, the process is executed repeatedly. An increment incr (given by an expression) is added to the variable in each iteration.

[Logical constants]

`true`, `false`

(continued on next page)

[Operators and their priorities]

Type of operation	Operator	Priority
Unary operation	+, −, not	High
Multiplication, division	×, ÷, %	↕
Addition, subtraction	+, −	
Relational operation	>, <, ≥, ≤, =, ≠	
Logical product	and	
Logical sum	or	Low

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

Company names and product names appearing in the test questions are trademarks or registered trademarks of their respective companies. Note that the ® and ™ symbols are not used within.

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

Q1. Read the following description concerning hardware design, and then answer Subquestions 1 and 2.

(Note) In this question, symbols •, +, $\bar{X}$, and n_{10} signify logical product, logical sum, logical not (of X), and decimal value (of *n*), respectively.

Hardware engineers are studying a hardware combinatorial scheme for multiplication operation of two unsigned 2-bit integers; $Z = X * Y$. The multiplicand X is denoted by X_1X_2, here X_1 is the most significant bit. The multiplier Y is denoted by Y_1Y_2, here Y_1 is the most significant bit. It is obvious that the maximum value of the product is 9, so the product of these two integers can be stored into unsigned 4-bit integer. The product Z is denoted by $Z_1Z_2Z_3Z_4$, here Z_1 is the most significant bit. The following list shows all of the multiplication combinations:

Z	$Z_1Z_2Z_3Z_4$	possible combination of X * Y
0_{10}	0 0 0 0	0_{10} * any, any * 0_{10}
1_{10}	0 0 0 1	1_{10} * 1_{10}
2_{10}	0 0 1 0	2_{10} * 1_{10}, 1_{10} * 2_{10}
3_{10}	0 0 1 1	3_{10} * 1_{10}, 1_{10} * 3_{10}
4_{10}	0 1 0 0	2_{10} * 2_{10}
5_{10}	0 1 0 1	
6_{10}	0 1 1 0	2_{10} * 3_{10}, 3_{10} * 2_{10}
7_{10}	0 1 1 1	
8_{10}	1 0 0 0	
9_{10}	1 0 0 1	3_{10} *3_{10}

To design a combinatorial scheme for the multiplication operation, the truth table for each output Z_1, Z_2, Z_3 and Z_4 should be derived based on the above multiplication combinations.

Y_1Y_2 \ X_1X_2	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	0
10	0	0	0	0

Truth table for Z_1

Y_1Y_2 \ X_1X_2	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	1
10	0	0	1	1

Truth table for Z_2

Y_1Y_2 \ X_1X_2	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	0	1	0	1
10	0	1	1	0

Truth table for Z_3

Y_1Y_2 \ X_1X_2	00	01	11	10
00	0	0	0	0
01	0	1	1	
11	0	1		
10	0			

Truth table for Z_4

Figure 1 Truth tables for Z_1, Z_2, Z_3 and Z_4 (partially incomplete)

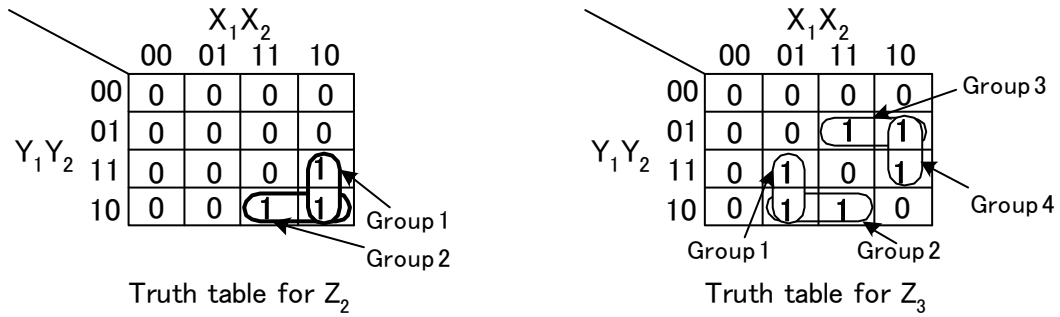


Figure 2 Detailed truth tables for Z_2 and Z_3

To derive the logic expression for each output Z_1, Z_2, Z_3 and Z_4, the Karnaugh Maps shown in Figures 1 and 2 have been constructed.

The logic expression for Z_1 is straightforward.

The truth table for Z_2 has two groups of 1s; Group 1 generates the term $X_1 \bullet \bar{X}_2 \bullet Y_1$, and Group 2 generates the term $X_1 \bullet Y_1 \bullet \bar{Y}_2$.

Therefore, the logic expression for Z_2 is $X_1 \bullet \bar{X}_2 \bullet Y_1 + X_1 \bullet Y_1 \bullet \bar{Y}_2$.

The truth table for Z_3 has four groups of 1s; Group 1 generates the term $\bar{X}_1 \bullet X_2 \bullet Y_1$, Group 2 generates the term $X_2 \bullet Y_1 \bullet \bar{Y}_2$, Group 3 generates the term $\boxed{\quad A \quad}$, and Group 4 generates the term $\boxed{\quad B \quad}$.

The logic expression for Z_4 can be derived similarly.

The following list summarizes the logic expressions for Z_1, Z_2, Z_3 and Z_4.

$$Z_1 = X_1 \bullet X_2 \bullet Y_1 \bullet Y_2$$

$$Z_2 = X_1 \bullet \bar{X}_2 \bullet Y_1 + X_1 \bullet Y_1 \bullet \bar{Y}_2$$

$$Z_3 = \bar{X}_1 \bullet X_2 \bullet Y_1 + X_2 \bullet Y_1 \bullet \bar{Y}_2 + \boxed{\quad A \quad} + \boxed{\quad B \quad}$$

$$Z_4 = \boxed{\quad C \quad}$$

Figure 3 shows the logic circuit that implements the logic expressions shown above.

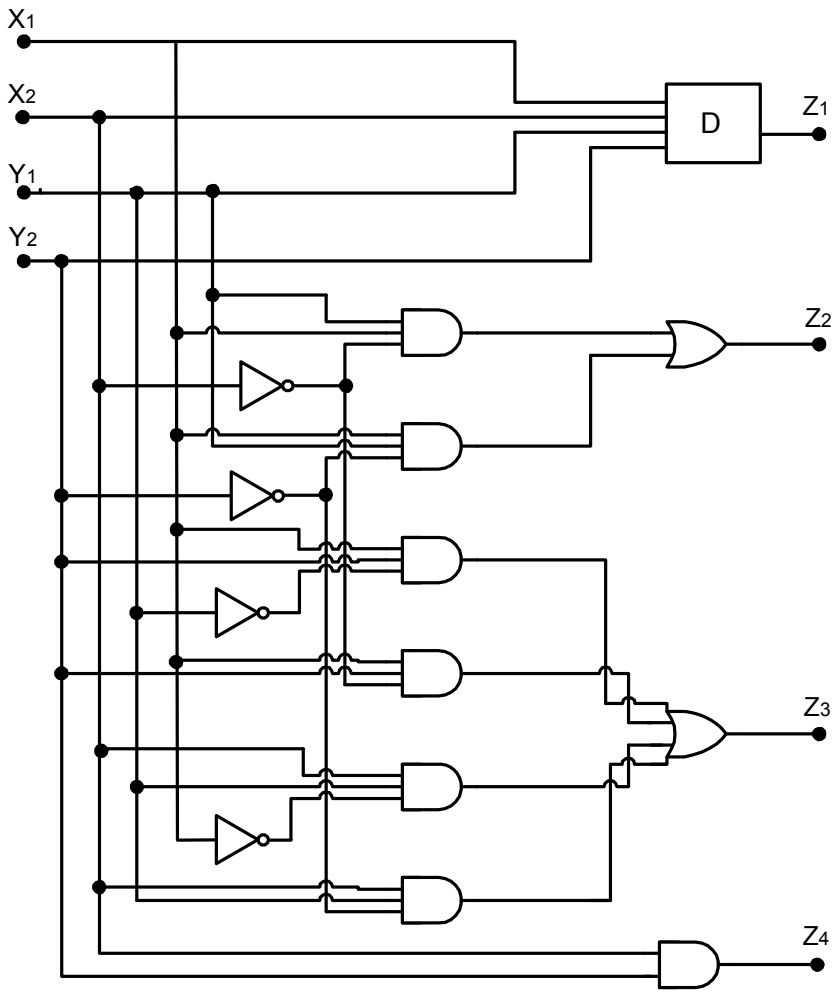


Figure 3 Logic circuit that implements the logic expressions

Subquestion 1

From the answer group below, select the correct truth table for Z_4 to complete the Figure 1.

Answer group

a)

Y_1Y_2 \ X_1X_2	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	0	0
10	0	0	0	0

b)

Y_1Y_2 \ X_1X_2	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

c)

Y_1Y_2 \ X_1X_2	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	1

d)

Y_1Y_2 \ X_1X_2	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	1
10	0	0	1	0

Subquestion 2

From the answer groups below, select the correct answer to be inserted into each blank [A] through [C] in the above description, and the blank [D] in Figure 3.

Answer group for A and B

a) $X_1 \bullet \bar{X}_2 \bullet Y_1$

b) $X_1 \bullet \bar{X}_2 \bullet Y_2$

c) $\bar{X}_1 \bullet X_2 \bullet Y_1$

d) $\bar{X}_1 \bullet X_2 \bullet Y_2$

e) $X_1 \bullet Y_1 \bullet \bar{Y}_2$

f) $X_2 \bullet Y_1 \bullet \bar{Y}_2$

g) $X_1 \bullet \bar{Y}_1 \bullet Y_2$

h) $X_2 \bullet \bar{Y}_1 \bullet Y_2$

Answer group for C

a) $X_1 \bullet X_2 \bullet Y_1 + \bar{Y}_1 \bullet Y_2$

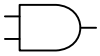
b) $\bar{X}_1 \bullet \bar{X}_2 \bullet Y_2 + X_2 \bullet Y_1 \bullet Y_2 + X_1 \bullet \bar{Y}_1 \bullet Y_2$

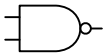
c) $X_1 \bullet Y_1 + \bar{X}_1 \bullet X_2 \bullet Y_1$

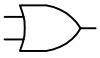
d) $X_2 \bullet Y_2$

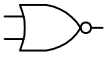
e) $\bar{X}_2 \bullet Y_2 + X_1 \bullet \bar{X}_2 \bullet Y_1$

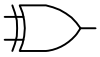
Answer group for D

a)  (AND gate)

b)  (NAND gate)

c)  (OR gate)

d)  (NOR gate)

e)  (Exclusive OR gate)

Q2. Read the following description concerning ASCII characters, and then answer Subquestion.

ASCII is a commonly used character-encoding scheme. Since ASCII is a 7-bit code, it can represent 128 characters: 94 graphic characters (codes 21_{16} – $7E_{16}$), 1 space character (code 20_{16}), and 33 control characters (codes 00_{16} – $1F_{16}$ and $7F_{16}$). The detailed ASCII code chart is shown in the table.

From the table, binary representation of ASCII characters can be obtained. For example, binary representation of "A" is:

b6	b5	b4	b3	b2	b1	b0		binary		hexadecimal		decimal
1	0	0	0	0	0	1	→	1000001_2	=	41_{16}	=	65_{10}

Table 128 ASCII characters

Least significant bits b3 b2 b1 b0	Most significant bits b6 b5 b4							
	000	001	010	011	100	101	110	111
0 0 0 0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	STX	DC2	"	2	B	R	b	r
0 0 1 1	ETX	DC3	#	3	C	S	c	s
0 1 0 0	EOT	DC4	$	4	D	T	d	t
0 1 0 1	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	ACK	SYN	&	6	F	V	f	v
0 1 1 1	BEL	ETB	'	7	G	W	g	w
1 0 0 0	BS	CAN	(	8	H	X	h	x
1 0 0 1	HT	EM	)	9	I	Y	i	y
1 0 1 0	LF	SUB	*	:	J	Z	j	z
1 0 1 1	VT	ESC	+	;	K	[	k	{
1 1 0 0	FF	FS	,	<	L	\	l	|
1 1 0 1	CR	GS	-	=	M	]	m	}
1 1 1 0	SO	RS	.	>	N	^	n	~
1 1 1 1	SI	US	/	?	O	_	o	DEL

Subquestion

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

Here, X and Y are 8-bit binary type variables. Operators and, or, and xor signify logical product, logical sum, and exclusive logical sum, respectively.

(1) When X contains one of ASCII numeric characters ('0' – '9'), binary integer value of X can be obtained in Y by executing the following expression

- Y ← [A]

(2) When the following ASCII character string is displayed or printed on a device which follows ASCII control characters, the result will be [B] .

1001001, 1010000, 1000101, 1000011, 0101010,
0001010, 0111001, 0001010, 0101010

(3) When X contains one of ASCII lower-case alphabet characters ('a' – 'z'), there are several methods to convert X into the upper-case alphabet characters ('A' – 'Z'). Three such methods are shown below.

(Method 1) • X ← X - [C]

(Method 2) • X ← X and [D]

(Method 3) • X ← X xor [E]

Answer group for A

a) X + 03_{16}
b) X - 30_{16}
c) X and 03_{16}
d) X or 30_{16}
e) X xor 03_{16}

Answer group for B

a) IPEC*!9!*
b) IPUC %9%
c) IPEC*
9
*
d) IPUC
*9
*

Answer group for C

a) 01_{16}
b) 02_{16}
c) 03_{16}
d) 10_{16}
e) 20_{16}
f) 30_{16}

Answer group for D and E

a) 03_{16}
b) 20_{16}
c) CF_{16}
d) DF_{16}
e) EF_{16}
f) FE_{16}

Q3. Read the following description concerning a database for laboratory reservation, and then answer Subquestions 1 and 2.

A university has developed a laboratory reservation system using a database. Figure 1 shows a part of the university's database.

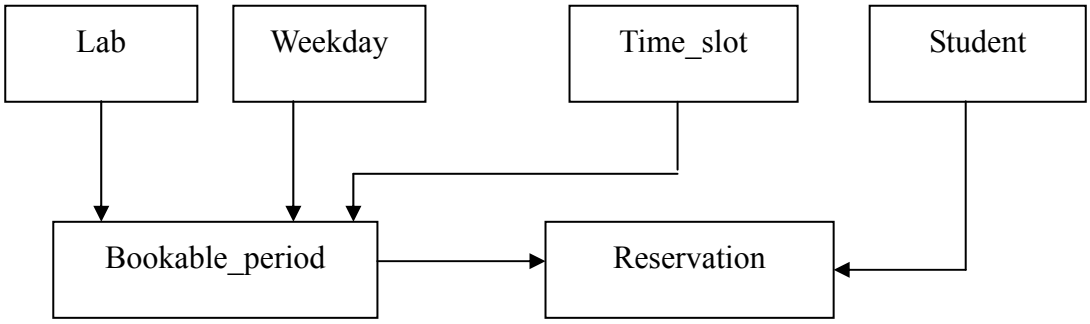


Figure 1 A part of the university's database

Contents of the 6 tables in Figure 1 are as follows.

(1) `Lab` table: Contains a list of computer labs of this university, including `lab_id`, `lab_address`, and `no_computers_in_total` (number of computers in total in the lab).

(2) `Weekday` table: Contains a list of weekdays of a week that students can study in the university. The values of `week_day` column are all available days of a current week, from Monday to Saturday.

(3) `Time_slot` table: Contains information about the time period that students can practice in the lab, including `slot_id`, `usage_time_from`, and `usage_time_to`. An example of the `Time_slot` table is shown below.

Slot_id	Usage_time_from	Usage_time_to
1	7h AM	8h 30 AM
2	8h 40 AM	10h 10 AM
3	10h 20 AM	11h 50 AM
4	...	...
5	...	...

(4) `Student` table: Contains information about all students, including `student_id`, `student_name`, `student_address`, and `student_phone_number`.

(5) `Bookable_period` table: Contains information about the period during a week for each lab that is available for reservation, including `lab_id`, `week_day`, and `slot_id`.

(6) `Reservation table`: Contains student's reservation records, including `lab_id`, `week_day`, `slot_id`, and `student_id`.

When a student wants to practice in any lab, he or she must reserve the lab in advance. The reservation procedure is as follows.

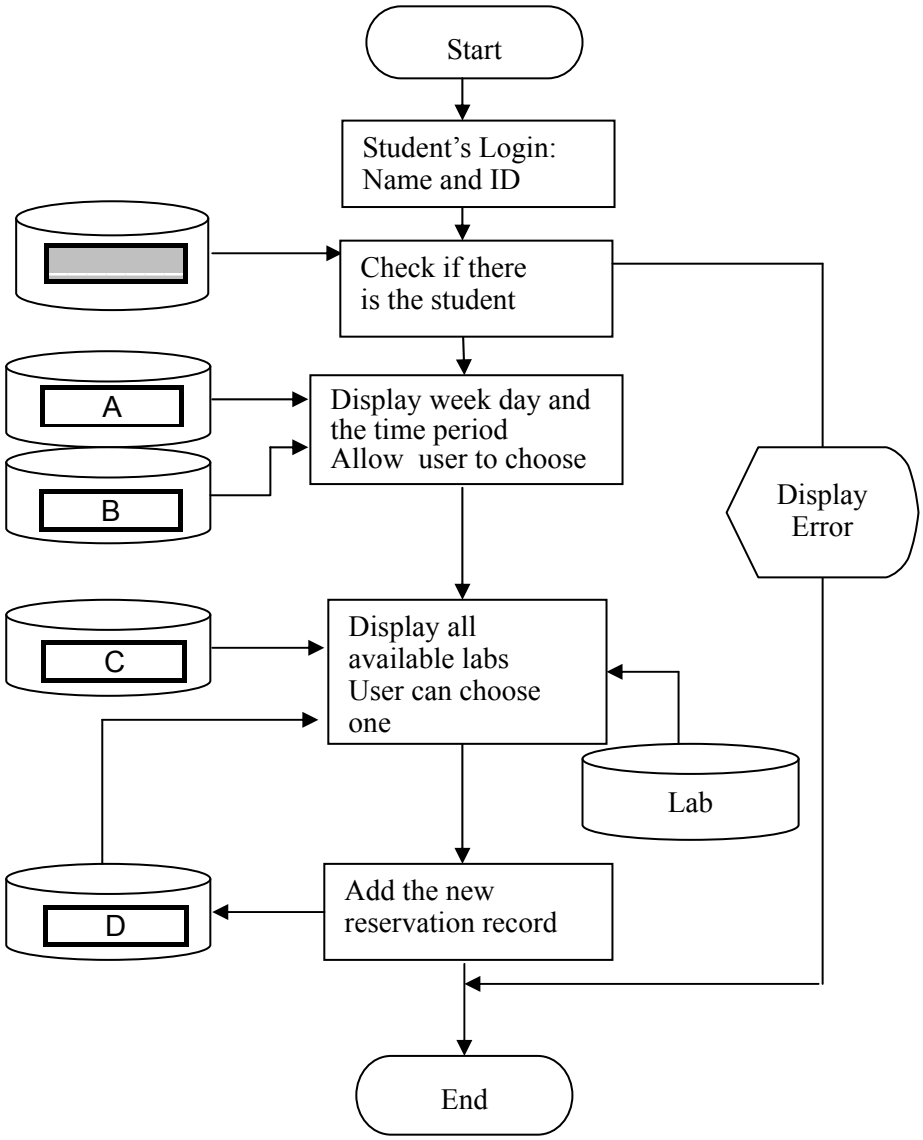
(1) Student must declare student ID and name. The system will check if the declared student exists in the database or not.

(2) If the student's information is stored in the system, the list of weekdays and time periods will be displayed for the student to choose.

(3) The student is asked to choose a time slot from the list. Only one time slot can be reserved.

(4) At the end, a new reservation record will be added in `Reservation` table.

The flowchart of the reservation procedure is shown in Figure 2.



Note: Shaded part is not shown

Figure 2 Flowchart of the reservation procedure

Subquestion 1

From the answer group below, select the correct table name to be inserted into each blank [ ] in Figure 2.

Answer group

a) Bookable_period
b) Lab
c) Reservation
d) Student
e) Time_slot
f) Weekday

Subquestion 2

The following SQL statement displays all labs available for reservation. From the answer group below, select the correct answer to be inserted into each blank [ ] in the following SQL statement. Here, `:week_day` and `:slot_id` are host system variables

```
SELECT  Lab.lab_id, Lab.lab_address
  FROM  Lab, Bookable_period
  WHERE Lab.lab_id = Bookable_period.lab_id
    AND Bookable_period.week_day = :week_day
    AND Bookable_period.slot_id = :slot_id
    AND Lab.lab_id [   E   ]
        (SELECT Lab.lab_id
           FROM Reservation, Lab
           WHERE Reservation.lab_id = Lab.lab_id
             AND Reservation.week_day = :week_day
             AND Reservation.slot_id = :slot_id
           GROUP BY [   F   ]
           HAVING [   G   ] )
    ORDER BY Lab.lab_id
```

Answer group

a) `COUNT(Reservation.student_id) < no_computers_in_total`
b) `COUNT(Reservation.student_id) = no_computers_in_total`
c) `IN`
d) `Lab.lab_id`
e) `Lab.lab_id, no_computers_in_total`
f) `NOT IN`
g) `SUM(Reservation.student_id) = no_computers_in_total`

Q4. Read the following description concerning user authentication, and then answer Subquestions 1 and 2.

Company *X* is studying the methods of user authentication to enable remote login from an external terminal to its in-house server. The method of transmitting the user ID and password to the server is used inside the company, and the safety of the three methods described below is being considered, including the enhancement of the password.

[Method 1: User ID and password method]

The terminal transmits the user ID and password entered by the user to the server. The server searches for the registered password based on the user ID, compares it with the transmitted password, and then returns a response allowing or denying the login. Fig. 1 shows the user ID and password method.

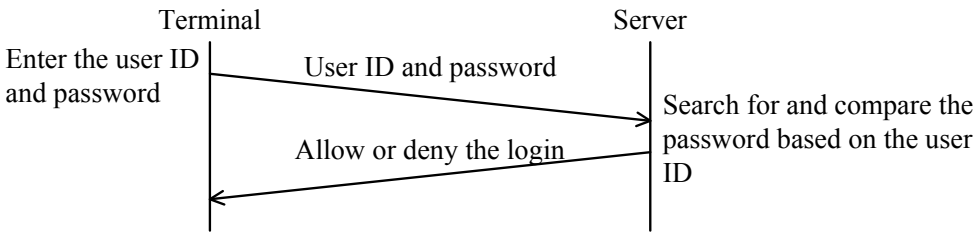


Fig. 1 User ID and password method

[Method 2: Challenge-response method]

The terminal transmits the user ID entered by the user to the server. When the server receives the user ID, the server transmits a randomly generated value c, called a challenge, to the terminal. The terminal calculates the hash value $h(p, c)$ from the password p entered by the user and the challenge c, and transmits it to the server as the response value. Then, the server searches for the registered password p' based on the user ID, calculates the hash value $h(p', c)$ by using the same hash function h as that of the terminal, compares it with the response value, and returns a response allowing or denying the login. Here, the hash function h is publicly known and can be calculated by any terminal. Fig. 2 shows the challenge-response method.

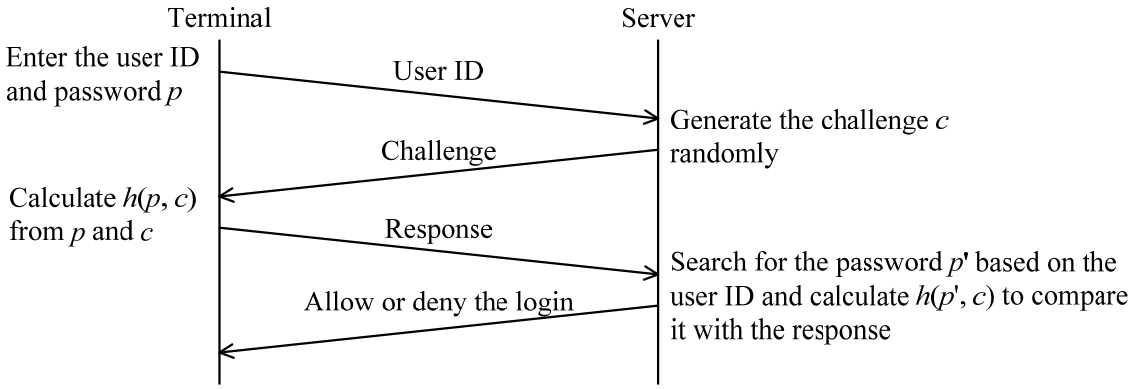


Fig. 2 Challenge-response method

[Method 3: Token (password generator) method]

The user is provided with a password generator, called a token, in which the user's user ID is registered. Fig. 3 shows an example of the token.

Fig. 3 Example of the token

The token has a timer built in it and, by using the function g, can generate and display the password $g(u, t)$ based on the user ID u and the time t. The user enters the user ID and the password generated and displayed by the token, and the terminal transmits them to the server. Then, the server generates the password $g(u, t)$ from the user ID u and the time t of the server by using the same function g as that of the token, compares it with the password received from the terminal, and returns a response allowing or denying the login.

It is guaranteed that the time of the token is synchronized with the time of the server. A certain amount of time is allowed as the delay from the password display by the token to the password generation by the server. Fig. 4 shows the token method.

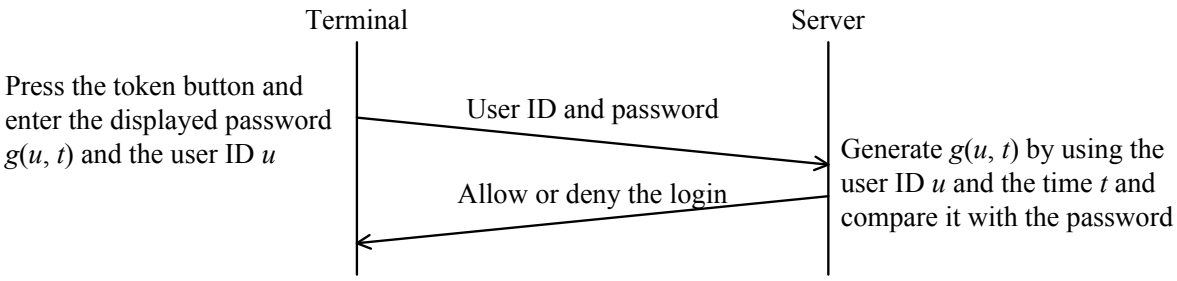


Fig. 4 Token method

Subquestion 1

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

Under methods 1 and 2, the user sets the password. When either of these methods is adopted, it is necessary to have the user set a password that is not easily guessed or, in other words, a password that is sufficiently robust.

Possible ways to increase the robustness of the password include making the password longer and using more different types of characters in the password. For example, when the maximum time required to find an 8-character password consisting of only 26 lowercase alphabetic characters in a brute force fashion is 1, the maximum required time is [A] if the length of the password is extended to 10 characters. Also, even when the length of the password remains 8 characters, the maximum required time is [B] if uppercase alphabetic characters are used as well.

Answer group

a) 1.25	b) 2	c) 208	d) 256
e) 260	f) 676	g) 1,024	

Subquestion 2

From the answer group below, select the correct answer to be inserted into each blank [ ] in the following description concerning the risk of the wiretapping. If needed, select the same answer more than once.

Depending on the user authentication method used, unauthorized login may occur by using illicitly obtained information (for example, user ID and password) as is.

(1) If the transmitted data is wiretapped on a communication path from outside the company, it is under [C] that unauthorized login to the server may occur anytime by using the stolen information as is, unless the user changes the password. Here, the communication path is not encrypted.

(2) If a program that reads keyboard input and transmits it to a third party is running on the terminal used for remote login from outside the company, it is under [D] that unauthorized login to the server may occur anytime by using the stolen information as is, unless the user changes the password.

(3) If a connection to an unauthorized server is made by mistake and the normal login operation is performed for the server, there is a possibility that information transmitted from a terminal may be stolen on that mistakenly connected server. It is under [E] that unauthorized login to the server may occur anytime by using the stolen information as is, unless the user changes the password.

Answer group

a) only method 1 b) only method 2 c) only method 3
d) only methods 1 and 2 e) only methods 1 and 3 f) only methods 2 and 3
g) all of methods 1, 2, and 3

Q5. Read the following description concerning program design, and then answer Subquestions 1 and 2.

A police department is planning to develop a traffic violation ticketing system. A traffic violation is detected by an automatic camera or by a policeman (hereinafter, enforcer). The traffic violation ticketing system consolidates a camera-based system and an enforcer-based system. The system sends a registered mail of the corresponding official violation ticket to the driver's address. In the absence of information about the driver, the violation ticket is sent to the address of the vehicle's owner.

[Explanation for Camera-Based Violation File]

A violation detecting system using cameras is installed on major intersections and highways. It captures speeding and red-light violations. Violations are recorded in the camera-based violation file as shown in Figure 1. The coordinate indicates the location of the camera. Multiple cameras on the same intersection use the same coordinate.

Coordinate	Vehicle Plate No	Vehicle Color	Vehicle Type	Date	Time	Violation Code

Figure 1 Format of the camera-based violation file

[Explanation for Enforcer-Based Violation File]

An enforcer makes use of a hand-held device to record violations. It makes use of the same coordinate system as the camera-based. Violations are recorded by the enforcer with the same fields in Figure 1 plus the fields shown in Figure 2.

(Same fields of Figure 1)	Driver Name	Driver License No

Figure 2 Format of the enforcer-based violation file

[Explanation for Vehicle Registration File]

Each vehicle undergoes annual registration. The vehicle registration file contains several fields but a shared file with limited fields as shown in Figure 3 is used for this processing. Both the registration no and the plate no is unique to every vehicle. The plate no refers to the vehicle plate no in the violation ticket.

Registration No	Plate No	Color	Make	Owner Name	Address

Figure 3 Format of the vehicle registration file

[Explanation for Driver License File]

Each driver is issued a license with a unique driver license no. The basic information for each license record to be used for this processing is shown in Figure 4.

Driver License No	Name	Age	Sex	Contact No	Address

Figure 4. Format of the driver license file

[Explanation for Violation Coding System]

Each violation has a 3-digit code. The first number classifies the offense to 0 – Grave; 2 – Serious; 6 – Minor. A grave offense is an offense that directly endangers others and could lead a large fine plus license revocation. A serious offense would result to a fine and a license suspension. A minor offense leads to a fine. The next two numbers are sequence numbers that corresponds to the individual offenses within the same classification.

[Program Description]

(1) Process 1 – Remove duplicates in the camera-based violation file

A consolidation process begins by removing duplicates. Violations detected by cameras with the same coordinate for the same vehicle within 30 seconds are considered as one violation. In this case, if the cameras record different violations, only the heavier offense is used. This process generates a clean camera-based violation file.

(2) Process 2 – Consolidation of Violation Records (Refer to Figure 5)

A consolidation process merges the clean camera-based violations and the enforcer-based violations. If an enforcer-based violation occurs within 5 minutes after a camera-based violation, these violations are considered as one violation and only the heavier offense is used. Otherwise, each violation is handled as an independent violation. This process generates a consolidated violation file.

(3) Process 3 – Generation of Offenses Mailing List (Refer to Figure 6)

A mailing list file is generated that should contain all the violation ticket information including the driver name and driver license no. The record would also contain the address of the driver as found in the driver license file. In cases where the driver is unknown (camera-based only), the owner of the vehicle is assumed to be the driver, and the owner name and address in the vehicle registration file will be used as the driver's name and address.

(4) Conditions and Assumptions

- The camera-based violation file and enforcer-based violation file are sorted in ascending order according to vehicle plate no, date, time, and coordinate.

- Process 1 is executed everyday. Processes 2 and 3 are executed twice a week.
- Date and time can be compared for <, = and >.
- Calculate TimeLapse will do a lapse of time from camera-based violation to enforcer-based violation. The result will be in minutes.

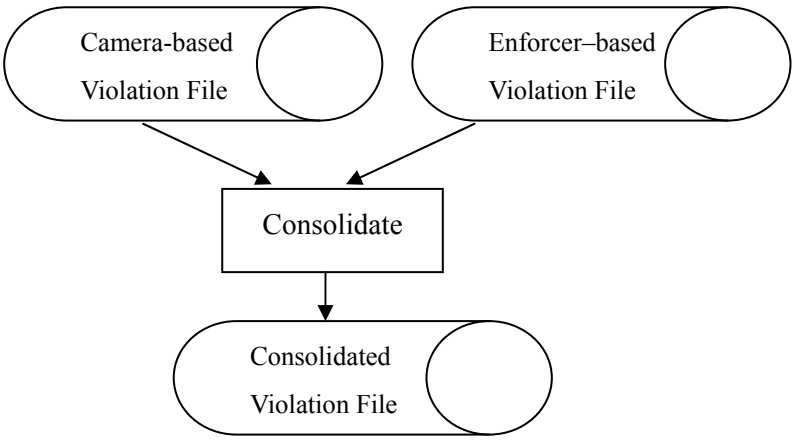


Figure 5 Input/Output Relational Diagram for Process 2

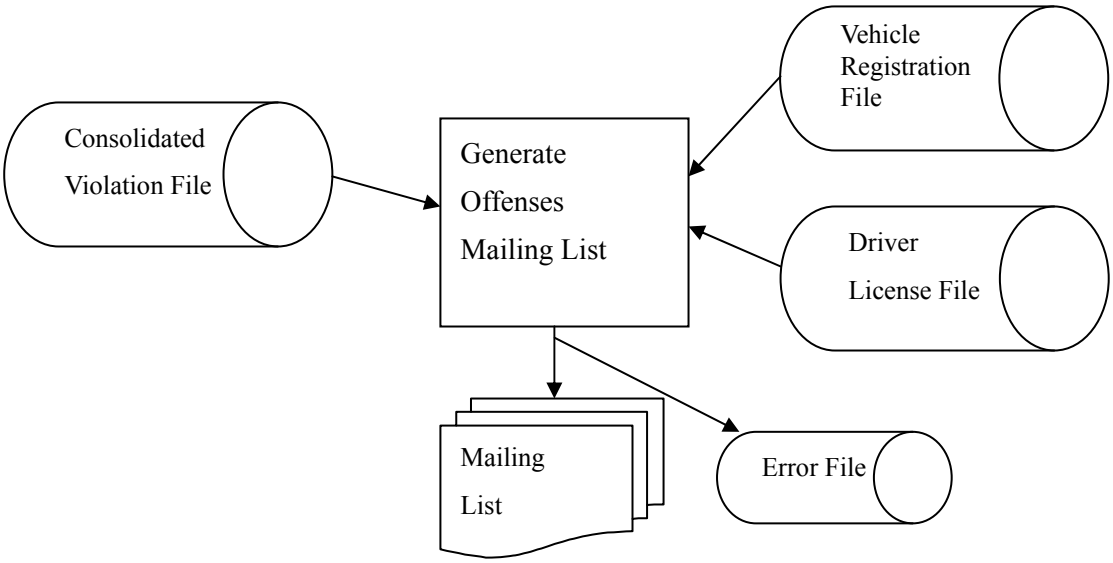
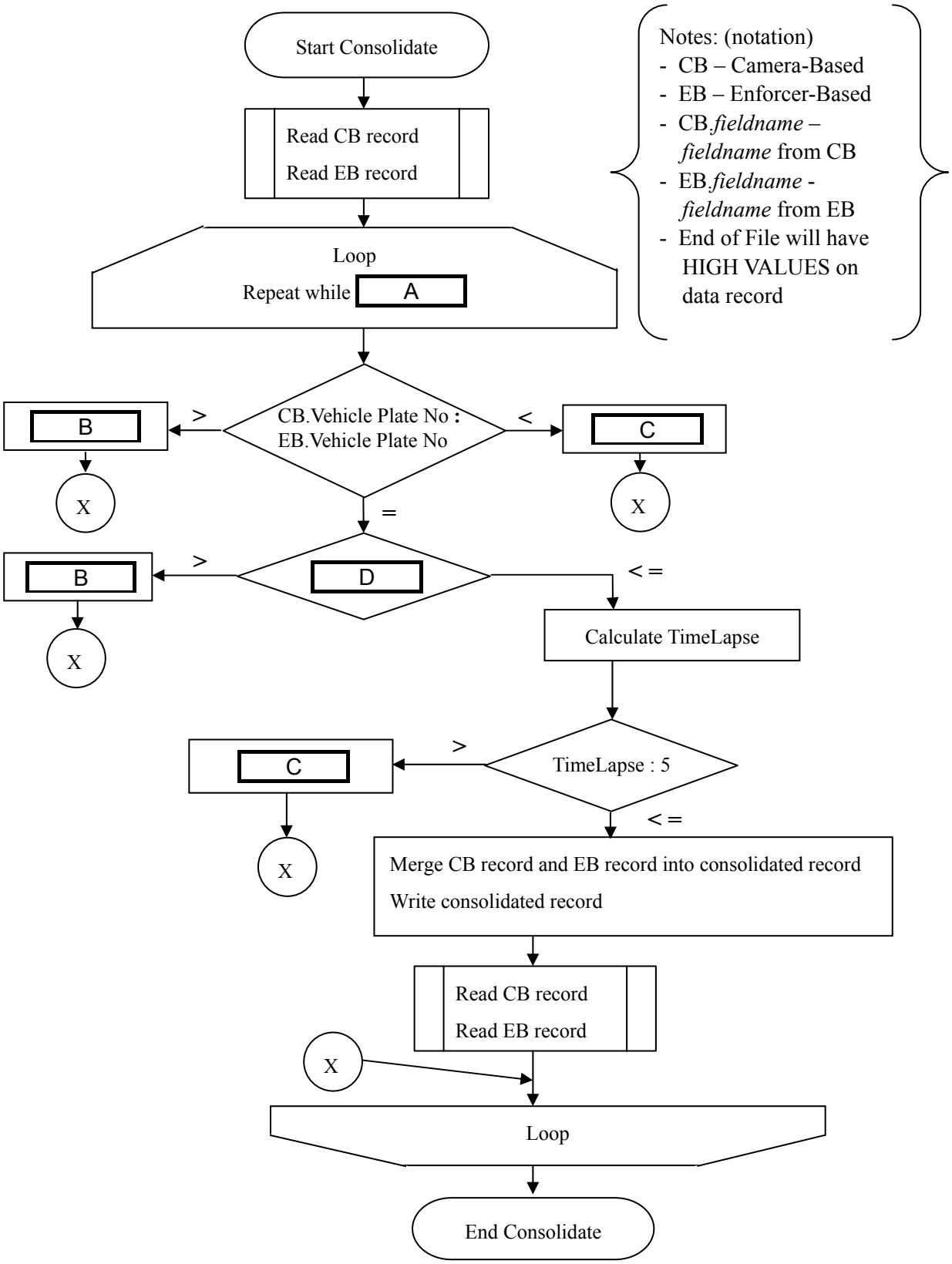


Figure 6 Input/Output Relational Diagram for Process 3

Subquestion 1

The following flowchart shows the program "Consolidate" for Process 2 – consolidation of violation records (Refer to Figure 5). From the answer groups below, select the correct answer to be inserted into each blank ☐ in the following flowchart.


Start Consolidate
Read CB record
Read EB record
Loop
Repeat while
A
Notes: (notation)
- CB – Camera-Based
- EB – Enforcer-Based
- CB.*fieldname* – *fieldname* from CB
- EB.*fieldname* - *fieldname* from EB
- End of File will have HIGH VALUES on data record
CB.Vehicle Plate No :
EB.Vehicle Plate No
>
<
=
B
C
X
X
B
D
>
<=
X
Calculate TimeLapse
C
>
TimeLapse : 5
<=
X
Merge CB record and EB record into consolidated record
Write consolidated record
Read CB record
Read EB record
X
Loop
End Consolidate

Answer group for A

a) CB.EOF = ‘No’ AND EB.EOF = ‘No’
b) CB.EOF = ‘No’ OR EB.EOF = ‘No’
c) CB.EOF = ‘Yes’ AND EB.EOF = ‘Yes’
d) CB.EOF = ‘Yes’ OR EB.EOF = ‘Yes’

Answer group for B and C

a) Move CB record to consolidated record
   Write consolidated record
   Read CB record
b) Move CB record to consolidated record
   Write consolidated record
   Read EB record
c) Move EB record to consolidated record
   Write consolidated record
   Read CB record
d) Move EB record to consolidated record
   Write consolidated record
   Read EB record
e) Move CB record to consolidated record
   Write consolidated record
   Read CB record
   Move EB record to consolidated record
   Write consolidated record
   Read EB record

Answer group for D

a) CB.Coordinate : EB.Coordinate
b) CB.Date + CB.Time : EB.Date + EB.Time
c) CB.Time : EB.Time
d) EB.Coordinate : CB.Coordinate
e) EB.Date + EB.Time : CB.Date + CB.Time
f) EB.Time : CB.Time

Subquestion 2

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

Process 3 uses the consolidated violation file, vehicle registration file and driver license file to generate the mailing list file (Refer to Figure 6). The format of the consolidated violation file will be identical to the enforcer-based violation file.

Process 3 makes use of data redundancy in the records from three input files to allow for verification. Stolen or fake plate numbers and driver licenses would be flagged during the generation of the mailing list file and will be included in the error file. To make this verification, after reading the driver license record, the field [E] in the consolidated record should be identical to the field [F] in the driver license record.

Answer group

a) Address
b) Driver License No
c) Driver Name
d) Name
e) Owner Name
f) Plate No
g) Vehicle Plate No

Q6. Read the following description of a program and the program itself, and then answer Subquestions 1 through 3.

[Program Description]

This program is an implementation of Floyd-Warshall algorithm, which finds the shortest path lengths for all the pairs of vertices in a weighted graph.

A graph is composed of vertices and edges. For example, the graph in Figure 1 is composed of 6 vertices and 8 edges.

A graph with n vertices ($n \geq 2$) can be represented by a 2-dimensional array sized $n \times n$. In this program, 2-dimensional array `path` is used to define a graph and to find the shortest path lengths for pairs of vertices. An element `path[i][j]` represents the path length between vertices `i` and `j`. The value 10^6 expresses that currently no path is discovered for the pair of vertices. Assuming that a graph is undirected, i.e. `path[i][j]=path[j][i]`, and any given or discovered path length is non-negative, and it is less than 10^6.

Table 1.a shows the array representation of the graph in Figure 1.

The following procedure finds the shortest path lengths for all the pairs of vertices in a given graph. Indexes of array `path` start at 0.

(1) Initialize the array `path`. If there is an edge that directly connects vertices `i` and `j`, then set its path length to `path[i][j]`, otherwise, set 10^6 to `path[i][j]`. A path length from a vertex to itself is always 0.

(2) Set 1 to `k`. `k` is an index of a vertex which will be used as an intermediate vertex to find a shorter path.

(3) Update the current pass length in `path[i][j]` when `path[i][k]+path[k][j]` is shorter than `path[i][j]`.

(4) Add 1 to `k`. If `k` is less than or equal to `n`, go to step (3).

(5) The shortest path lengths for pairs of vertices are obtained in the array `path`.

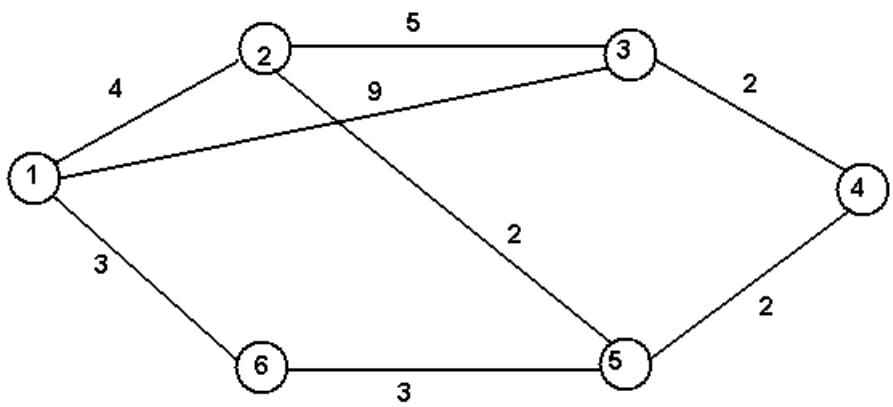


Figure 1 Example of a weighted graph

Table 1 shows the execution results of the procedure for the graph in Figure 1.

Table 1 Execution results of the procedure

Table 1.a After step (1)

i \ j	1	2	3	4	5	6
1	0	4	9	10^6	10^6	3
2	4	0	5	10^6	2	10^6
3	9	5	0	2	10^6	10^6
4	10^6	10^6	2	0	2	10^6
5	10^6	2	10^6	2	0	3
6	3	10^6	10^6	10^6	3	0

Note:

(1) Table 1.a shows the initial contents

(2) Table 1.g shows the final contents

(3) Updated elements are shaded.

Table 1.b After step (3) when k=1

i \ j	1	2	3	4	5	6
1	0	4	9	10^6	10^6	3
2	4	0	5	10^6	2	7
3	9	5	0	2	10^6	12
4	10^6	10^6	2	0	2	10^6
5	10^6	2	10^6	2	0	3
6	3	7	12	10^6	3	0

Table 1.c After step (3) when k=2

i \ j	1	2	3	4	5	6
1	0	4	9	10^6	6	3
2	4	0	5	10^6	2	7
3	9	5	0	2	7	12
4	10^6	10^6	2	0	2	10^6
5	6	2	7	2	0	3
6	3	7	12	10^6	3	0

Table 1.d After step (3) when k=3

i \ j	1	2	3	4	5	6
1	0	4	9	11	6	3
2	4	0	5	7	2	7
3	9	5	0	2	7	12
4	11	7	2	0	2	14
5	6	2	7	2	0	3
6	3	7	12	14	3	0

Table 1.e After step (3) when k=4

i \ j	1	2	3	4	5	6
1						
2						
3			A			
4						
5						
6						

Table 1.f After step (3) when k=5

i \ j	1	2	3	4	5	6
1	0					
2		0		(not shown)		
3			0			
4				0		
5	(not shown)				0	
6						0

Table 1.g After step (3) when k=6

i \ j	1	2	3	4	5	6
1	0	4	9	8	6	3
2	4	0	5	4	2	5
3	9	5	0	2	4	7
4	8	4	2	0	2	5
5	6	2	4	2	0	3
6	3	5	7	5	3	0

[Program]

```
○ Procedure ShortestPath
○ Integer: i, j, k, n, path[1000][1000]

• read n                          /* Read number of vertices into n */
■ i: 1, i<=n, 1                   /* Loop for clearing diagonal elements */
  • path[i][i] ← 0
■
■ i: [      B      ]              /* Loop for reading data into          */
  ■ j: [      C      ]            /*     upper-right triangle of the array */
    • read path[i][j]             /* Read path length between vertices i and j */
    • path[j][i] ← path[i][j]
  ■
  • print path                    /* Print the initial contents of array path */
■

■ k: 1, k<=n, 1
  ■ i: [      B      ]            /* Loop for reading data into          */
    ■ j: [      C      ]          /*     upper-right triangle of the array */
      ▲ path[i][j] > (path[i][k]+path[k][j])
      • path[i][j] ← path[i][k]+path[k][j]
      • path[j][i] ← path[i][j]
      ▼
    ■
  ■
  • print k,path                  /* Print k and the contents of array path */
■
/* End of the procedure ShortestPath */
```

Subquestion 1

From the answer group below, select the correct answer to be inserted into the blank [ ] in Table 1.e.

Answer group

a)

0	4	9	11	6	3
4	0	5	7	2	5
9	5	0	2	4	12
11	7	2	0	2	14
6	2	4	2	0	3
3	5	12	14	3	0

b)

0	4	9	11	6	3
4	0	5	7	2	5
9	5	0	2	4	12
11	7	2	0	2	14
6	2	4	2	0	3
3	5	12	14	3	0

c)

0	4	9	11	6	3
4	0	5	7	2	7
9	5	0	2	4	7
11	7	2	0	2	14
6	2	4	2	0	3
3	7	7	14	3	0

d)

0	4	9	11	6	3
4	0	5	7	2	7
9	5	0	2	4	12
11	7	2	0	2	14
6	2	4	2	0	3
3	7	12	14	3	0

Subquestion 2

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

Answer group

a) `1, i<=n-1, 1`
b) `1, i<=n, 1`
c) `i, j<=n-1, 1`
d) `i+1, j<=n, 1`
e) `k, j<=n-1, 1`
f) `k+1, i<=n, 1`

Subquestion 3

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

The amount of resources required to solve instances depends on the size of instances. As for the given algorithm, the time required to solve instances (time complexity) can be expressed as [D], and the memory space required to solve instances (space complexity) can be expressed as [E]. Here, n represents the number of vertices.

Answer group

a) $O(2^n)$
b) $O(n \log n)$
c) $O(n)$
d) $O(n^2)$
e) $O(n^3)$
f) $O(n^4)$

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

Q7. Read the following description of a C program and the program itself, and then answer Subquestions 1 and 2.

[Program Description]

The function `eval` calculates the value of an expression written in Prefix notation.
In Prefix notation, operators are written before their operands. For example, an expression `A * ( B + C ) / D` is equivalent to an expression in Prefix notation `/ * A + B C D`.
To simplify the program, assuming that:

- given expression is always valid.
- given expression does not contain trailing blanks.
- given expression does not cause division by 0.
- each number in an expression is a non-negative integer.

Specifications of the functions used in this program are as follows:

(1) Function `double eval (char s[], int *n)`

Arguments: `s` - string of an expression written in Prefix notation
`n` - pointer to the index of string `s`

Function: Calculates the value of the expression in string `s`, starting from the index position `n`.

Return value: The value of the expression in string `s`.

(2) Function `int seekChars(char s[], int *n)`

Arguments: `s` - string of an expression written in Prefix notation
`n` - pointer to the index of string `s`

Function: Starting from the index position `n`, seeks the first index position where the character is digit or operator (+, -, *, or /).

Return value: Returns 1 if found, 0 if not found. The value `n` indicates the first index position where the character is digit or operator (+, -, *, or /).

(3) Function `int stringToInteger(char s[], int *n)`

Arguments: `s` - string of an expression written in Prefix notation
`n` - pointer to the index of string `s`

Function: Convert the sub-string starting from the index position `n` to integer.

Return value: Returns the integer converted from that sub-string. The value `n` indicates the first index position where the character is not digit.

(4) Library function `int isdigit(int c)`

Return value: Returns non-zero (true) if `c` is a decimal digit, or 0 (false) otherwise.

[Program]

```
#include <conio.h>
#include <ctype.h>
#include <stdio.h>
#include <string.h>

double eval(char s[], int *n);
int seekChars(char s[], int *n);
int stringToInteger(char s[], int *n);

void main()
{
   char s[80];
   int n = 0;
   double result;

   strcpy(s, "- * 3 + 5 10 17");
   result = eval(s, &n);
   printf("\n%s = %.2lf", s, result);
   getch();
}

double eval(char s[], int *n)
{
  char ch;

   if (seekChars(s, n) == 1)  [    A    ];
   else return 0;
   if(isdigit(ch))  return stringToInteger(s, n);
   else
   {
      [    B    ];
      switch(ch)
      {
         case '+': return(eval(s, n) + eval(s, n)) ;
         case '-': return(eval(s, n) - eval(s, n)) ;
         case '*': return(eval(s, n) * eval(s, n)) ;
         case '/': return(eval(s, n) / eval(s, n)) ;
      }
   }
}
```

```
int seekChars(char s[], int *n)
{
   if (s[*n] == '\0') return 0;
   while ( [   C   ]
          && s[*n] != '+' && s[*n] != '-'
          && s[*n] != '*' && s[*n] != '/')
      [   B   ];
   if (s[*n] == '\0')  return 0;
   else return 1;
}

int stringToInteger(char s[], int *n)
{
   int value = [   D   ];

   [   B   ];
   while(isdigit(s[*n]))
   {
      value = value * 10 + ([   D   ]);
      [   B   ];
   }
   return value;
}
```

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 and D

a) `ch = s[n]`
b) `ch = s[*n]`
c) `s[n]`
d) `s[n] - '0'`
e) `s[*n]`
f) `s[*n] - '0'`

Answer group for B

a) `n++`
b) `(&n)++`
c) `(*n)++`
d) `*n++`

Answer group for C

a) `isdigit(s[n])`
b) `isdigit(s[*n])`
c) `!isdigit(s[n])`
d) `!isdigit(s[*n])`

Subquestion 2

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

Concerning the behavior of the program when array `s` contains "`-*3+5 10 17`", the number of times that function `eval` is called is [E]. At the beginning of the sixth time, the index pointed by `n` is [F], if only one space character is used to separate integers (that is, there is one space character between integers `5` and `10` and between integers `10` and `17`).

Answer group

a) 2	b) 3	c) 4
d) 5	e) 6	f) 7
g) 8	h) 9	

Q8. Read the following description of a Java program and the program itself, and then answer Subquestion.

[Program Description]

The program reads student data from a text file, calculates the course grades using that data, and displays the results. Classes `Student`, `GraduateStudent`, `UndergraduateStudent`, and `ComputeGrades` are used.

`Student` is an abstract class representing a student. It stores the information about a student's grade and scores of three tests. Its specification is given below:

- Constructor `Student` creates a `Student` with no student data.
- Method `getCourseGrade` returns the course grade.
- Method `getName` returns the name.
- Method `getTestScore` returns the test score of the given test number.
- Method `setName` sets this `Student`'s name to the given name.
- Method `setTestScore` sets the test score of the given test number.
- Abstract method `computeCourseGrade` computes the course grade from the test scores. The formulas for the grade calculation are given by subclass implementation.

Class `GraduateStudent` is a subclass of `Student` with the `computeCourseGrade` implementation for graduate students.

Class `UndergraduateStudent` is a subclass of `Student` with the `computeCourseGrade` implementation for undergraduate students.

Class `ComputeGrades` will make use of the three classes given above (`Student`, `GraduateStudent` and `UndergraduateStudent`). This `ComputeGrades` class will read students data from a text file, compute the course grades and display the results. The format of the input file is given below:

Column	Description
1	Student type: `G` for graduates, `U` for undergraduates
2	Student first name
3	Student last name
4	Score of test-1
5	Score of test-2
6	Score of test-3

The columns are separated by one or more whitespace characters. Sample data from the text file and its execution results are as follows.

Sample data:
```
U  John Doe  76  65  75
G  Jill Hil  78  69  78
```

Execution results:
```
John Doe  76  65  75  Pass
Jill Hil  78  69  78  No Pass
```

[Program]

```
import java.io.FileReader;
import java.io.BufferedReader;
import java.io.IOException;

public abstract class Student {
   protected final static int NUM_OF_TESTS = 3;
   protected String name;
   protected int[] test;
   protected String courseGrade;

   abstract public void computeCourseGrade();

   public Student() {
      test = new int[NUM_OF_TESTS];
      courseGrade = "****";
   }

   public String getCourseGrade() {
      return courseGrade;
   }

   public String getName() {
      return name;
   }

   public int getTestScore(int testNumber) {
      return test[testNumber];
   }

   public void setName(String newName) {
      name = newName;
   }

   public void setTestScore(int testNumber, int testScore) {
      test[testNumber] = testScore;
   }
}
```

```
public class GraduateStudent extends Student {
   public void computeCourseGrade() {
      int total = 0;
      for (int i = 0; i < NUM_OF_TESTS; i++) {
         total += test[i];
      }
      if (total / NUM_OF_TESTS >= 80) {
         courseGrade = "Pass";
      } else {
         courseGrade = "No Pass";
      }
   }
}

public class UndergraduateStudent extends Student {
   public void computeCourseGrade() {
      int total = 0;
      for (int i = 0; i < NUM_OF_TESTS; i++) {
         total += test[i];
      }
      if (total / NUM_OF_TESTS >= 70) {
         courseGrade = "Pass";
      } else {
         courseGrade = "No Pass";
      }
   }
}

public class ComputeGrades {
   private static final int DEFAULT_SIZE = 25;
   private static final String UNDER_GRAD = "U";
   private static final String GRAD = "G";
   private Student[] roster;
   private int studentCount;

   public ComputeGrades() {
      this(DEFAULT_SIZE);
   }

   public ComputeGrades(int arraySize) {
      roster = new [      A      ][arraySize];
      studentCount = 0;
   }
```

```
public void processData(String file) {
   try {
      [   B   ];
      [   C   ];
      [   D   ];
   } catch (IOException ioe) {
      System.err.println("File Input Error");
   }
}

private void buildRoster(String filename) throws IOException {
   BufferedReader bufReader = null;
   try {
      bufReader = new BufferedReader(new FileReader(filename));
      String inputLine;
      while (studentCount < roster.length &&
             (inputLine = bufReader.readLine()) != null) {
         Student student = createStudent(inputLine);
         if (student != null) {
            roster[studentCount++] = student;
         }
      } // while
   } finally {
      if (bufReader != null) bufReader.close();
   }
}

private void computeGrade() {
   for (int i = 0; i < studentCount; i++) {
      [   E   ].computeCourseGrade();
   }
}
```

```
   private Student createStudent(String line) {
      String[] tokens = line.split("\\s+", Student.NUM_OF_TESTS + 4);
      if (tokens.length < Student.NUM_OF_TESTS + 3) {
         return null;
      }
      Student student;
      if (tokens[0].equals(UNDER_GRAD)) {
         student = new [      F      ]();
      } else if (tokens[0].equals(GRAD)) {
         student = new [      G      ]();
      } else {
         return null;
      }
      //set the student name
      student.setName(tokens[1] + " " + tokens[2]);
      try {
         //set the student test scores
         for (int testNum = 0;
              testNum < Student.NUM_OF_TESTS; testNum++) {
            student.setTestScore(testNum,
                                 Integer.parseInt(tokens[testNum + 3]));
         }
      } catch (NumberFormatException e) {
         student = null;
      }
      return student;
   }

   private void printResult() {
      for (int i = 0; i < studentCount; i++) {
         //print one student
         System.out.print(roster[i].getName());
         for (int testNum = 0;
              testNum < Student.NUM_OF_TESTS; testNum++) {
            System.out.print("\t" + roster[i].getTestScore(testNum));
         }
         System.out.println("\t" + roster[i].getCourseGrade());
      }
   }
}
```

```
public class Main{
   public static void main(String[] args) {
      ComputeGrades gradeComputer = new ComputeGrades();
      gradeComputer.processData(args[0]);
   }
}
```

Subquestion

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

Answer group for A, F and G

a) `ComputeGrades`
b) `GraduateStudent`
c) `roster`
d) `student`
e) `Student`
f) `UndergraduateStudent`

Answer group for B, C and D

a) `buildRoster(file)`
b) `computeGrade()`
c) `ComputeGrades()`
d) `createStudent(file)`
e) `printResult()`
f) `processData(file)`

Answer group for E

a) `((GraduateStudent)roster[i])`
b) `((UndergraduateStudent)roster[i])`
c) `GraduateStudent[i]`
d) `roster[i]`
e) `Student[i]`
f) `UndergraduateStudent[i]`