

[Explanation of the Pseudo-Code Description Format]

Pseudo-Language Syntax	Description
[A continuous area where declarations and processes are described.
○	Declares names, types, etc. of procedures, variables, etc.
▪ Variable ← Expression	Assigns the value of an Expression to a Variable.
↑ Conditional expression ▪ Process 1 ─── ▪ Process 2 ↓	A 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 ─── ■	A repetition process with the termination condition at the top. The Process is executed while the Conditional expression is True.

[Operator]

Operation	Operator	Priority
Unary operation	+ - not	High ↑ ↓ Low
Multiplication and division operation	* /	
Addition and subtraction operation	+ -	
Relational operation	> < >= <= =	
Logical product	and	
Logical sum Exclusive logical sum	or xor	

[Logic type constant]

true false

Questions 1 through 5 are all compulsory. Answer every question.

Q1. Read the following description of lists, and then answer Subquestions 1, 2 and 3.

The structure of the lists is as shown in Figure 1.

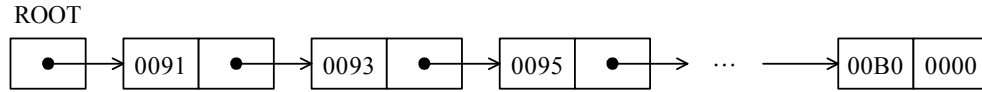


Fig. 1 List Structure

- 1) ROOT means the start of the list.
- 2) Elements of the list consist of two consecutive words. The first word stores the value, and the second word stores a pointer to the next element.
- 3) Each element of the list is linked to the next in ascending order of the value, and all values are unique. In the second word of the last element, 0000 is stored as a pointer.
- 4) A list with the structure shown in Fig. 1 is stored at addresses 00FF to 0117 in the main memory as shown in Figure 2. Address 00FF is ROOT.
- 5) One word consists of 16 bits, and addresses are assigned in word units.

Address	Contents	Address	Contents	Address	Contents	Address	Contents
∴	∴	0106	00A0	010E	00B0	0116	0099
00FF	0100	0107	010C	010F	0000	0117	0110
0100	0091	0108	00A9	0110	009B	0118	00A7
0101	010A	0109	0112	0111	0102	0119	0000
0102	009F	010A	0093	0112	00AB	011A	009C
0103	0106	010B	0104	0113	010E	011B	011C
0104	0095	010C		0114	00A2	011C	00A5
0105	0116	010D	0114	0115	0108	011D	0118

Fig. 2 State of the Main Memory

Subquestion 1

From the answer group below, select the correct answer for the contents of address 010C.

Answer group:

- a) 0099
- b) 00A1
- c) 00A3
- d) 00A4
- e) 00A5

Subquestion 2

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

In order to delete the elements at addresses 0110 and 0111, the contents of address A need to be changed to B.

Answer group:

- a) 0101
- b) 0102
- c) 0103
- d) 0104
- e) 0105
- f) 0113
- g) 0114
- h) 0115
- i) 0116
- j) 0117

Subquestion 3

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

In order to merge the sublist consisting of the three elements stored at addresses 0118 through 011D and the original list (before deleting the elements in Subquestion 2), the contents of address C need to be changed to 011A, the contents of address D to 0102, the contents of address E to 011C, and the contents of address F to 0108 respectively.

Answer group:

- a) 0109
- b) 010B
- c) 010D
- d) 010F
- e) 0111
- f) 0113
- g) 0115
- h) 0117
- i) 0119
- j) 011B

Q2. Read the following description of a relational database, and then answer Subquestions 1, 2 and 3.

A certain database consists of the following employee and department table. An employee works for a department and may or may not have a manager.

emp (employee) table

empno	ename	job	mgr	hiredate	sal	comm	deptno
-------	-------	-----	-----	----------	-----	------	--------

dept (department) table

deptno	dname	loc
--------	-------	-----

Subquestion 1

A display of the highest earner of each job is required, wherein the employees of each job category would be compared to the highest salary within the category.

ename	job	Highest-Salary
-------	-----	----------------

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

```
SELECT ename, job,  A
FROM emp
WHERE sal IN (SELECT  B FROM emp GROUP BY job)
ORDER BY  C descending;
```

Answer group:

- a) Highest-Salary
- b) MAX (sal)
- c) MAX (sal) Highest-Salary
- d) sal
- e) sal Highest-Salary
- f) salary

Subquestion 2

Provided that the empno is also used to denote mgr for another employee, what would be the appropriate select statement to determine the number of distinct managers without listing them.

No of managers

Answer group:

- a) select count (distinct (mgr)) "No of managers" from emp;
- b) select count (mgr (distinct)) "No of managers" from emp;
- c) select distinct (count (mgr)) "No of managers" from emp;
- d) select distinct (mgr) "No of managers" from emp;
- e) select mgr (distinct) "No of managers" from emp;

Subquestion 3

There is another table called salgrade, which has the salary grading and the lowest and highest salary within the grade.

salgrade table with data

grade	lowsal	highsal
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

A listing of employees including employee name, job, salary and salary grade is done to determine all employees in grade 2.

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

```
select e.ename, e.job, e.sal, s.grade
from  D ,
where  E and s.grade = 2;
```

Answer group for D:

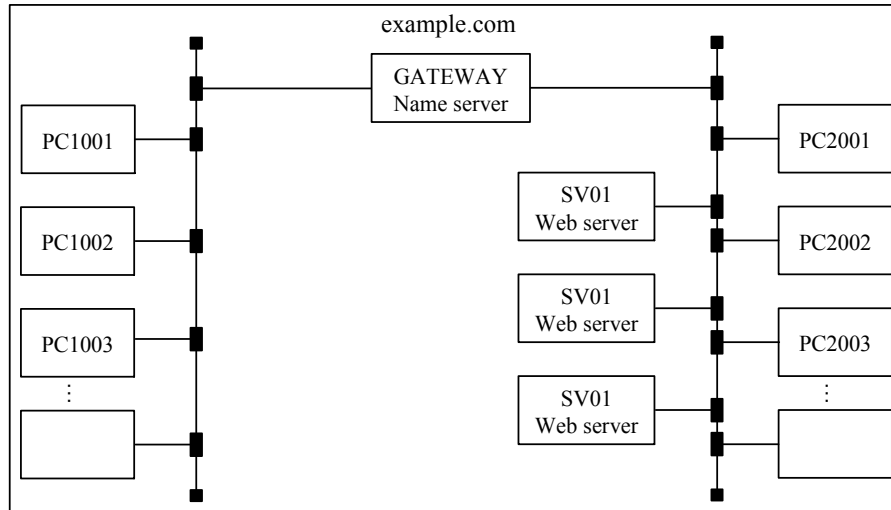
- a) e.emp, s.grade
- b) e.emp, s.salgrade
- c) emp e, grade s
- d) emp e, salgrade s
- e) emp, grade
- f) emp, salgrade

Answer group for E:

- a) e.sal <= s.highsal
- b) e.sal <= s.lowsal
- c) e.sal >= s.highsal
- d) e.sal >= s.lowsal
- e) e.sal between s.lowsal and s.highsal
- f) e.sal between s.lowsal or s.highsal

Q3. Read the following description of DNS (Domain Name System), and then answer the Subquestions 1 and 2.

In a certain enterprise, each computer in the in-house network is controlled by using a domain name. The configuration of the network in this enterprise is shown below.



Correspondence between the domain name and the IP address of each computer is as shown in the table below.

Table Correspondence between domain names and IP addresses

Domain name	IP address
PC1001.example.com	172.16.0.1
PC1002.example.com	172.16.0.2
PC1003.example.com	172.16.0.3
⋮	⋮
PC2001.example.com	172.31.0.1
PC2002.example.com	172.31.0.2
PC2003.example.com	172.31.0.3
⋮	⋮

Domain name	IP address
GATEWAY.example.com	172.16.0.101 172.31.0.101
SV01.example.com	172.31.0.91
SV01.example.com	172.31.0.92
SV01.example.com	172.31.0.93

Subquestion 1

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

A name server has a definition file for searching for IP addresses from domain names. In this file, a name server is defined as follows:

```
<Domain name to be defined>. IN NS <Domain name of name server>.
```

Moreover, correspondence between a domain name and an IP address is defined as follows:

```
<Domain name>. IN A <IP address>
```

In the case where GATEWAY.example.com is a name server, some of the statements in the definition file that this name server has are shown below.

```
example.com.      IN NS .example.com.  
  
localhost.example.com. IN A 127.0.0.1  
PC1001.example.com.  IN A 172.16.0.1  
PC1002.example.com.  IN A 172.16.0.2  
PC1003.example.com.  IN A 172.16.0.3  
:  
GATEWAY.example.com. IN A 172.16.0.101  
GATEWAY.example.com. IN A   
  
PC2001.example.com. IN A 172.31.0.1  
PC2002.example.com. IN A 172.31.0.2  
PC2003.example.com. IN A 172.31.0.3  
:  
GATEWAY-1.example.com. IN A 172.16.0.101  
GATEWAY-2.example.com. IN A 172.31.0.101  
  
SV01.example.com.  IN A 172.31.0.91  
SV01.example.com.  IN A 172.31.0.92  
SV01.example.com.  IN A 172.31.0.93
```

Answer group:

- | | | |
|--------------|----------------|-----------------|
| a) 127.0.0.1 | b) 172.31.0.91 | c) 172.31.0.101 |
| d) GATEWAY | e) localhost | f) SV01 |

Subquestion 2

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

In a DNS, a client which queries with a name server is called a resolver.

If IP addresses belonging to different subnetworks are defined for the same domain name, then the name server returns, on a priority basis, the IP addresses that belong to the same subnetwork to which the resolver belongs. This mechanism is applied in the case of the definition of .example.com.

On the other hand, if different IP addresses belonging to the same subnetwork are defined for the same domain name, then the name server returns those IP addresses cyclically in the order of definition. This mechanism is called round robin, and is used to distribute accesses to the server. This mechanism is applied in the case of the definition of .example.com.

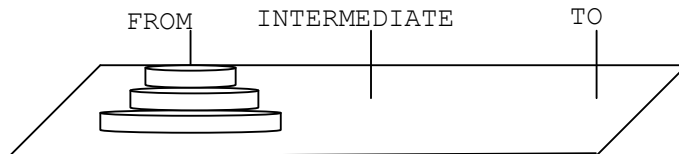
Answer group:

- a) GATEWAY-1
- b) GATEWAY-2
- c) GATEWAY
- d) localhost
- e) SV01

Q4. Read the following description of a program and the program itself, and then answer Subquestions 1, 2 and 3.

[Program Description]

This is a subprogram called `TowerHanoiGame` that move sequence of `n` disks in peg `FROM` to peg `TO` with the same rule.



1) Procedures are as follows:

The Towers of Hanoi game is an example of problem whose solution demands recursion. The game consists of a board with three vertical pegs labeled `FROM`, `INTERMEDIATE`, and `TO`, and sequence of `n` disks with holes in their centers. The radius of the disks are in an arithmetic progression (e.g., 5cm, 6cm, 7cm...) and are mounted on the peg `FROM`. The rule is that no disk may be above a smaller disk on the same peg. The objective of the game is to move all the disks from peg `FROM` to peg `TO`, one disk at a time, without violating the rule.

The general solution to the Tower of Hanoi game is naturally recursive:

Part 1: move the smaller `n-1` disks from peg `FROM` to peg `INTERMEDIATE`

Part 2: move the remaining disk from peg `FROM` to peg `TO`

Part 3: move the smaller `n-1` disks from peg `INTERMEDIATE` to peg `TO`

The first and third steps are recursive: apply the complete solution to `n-1` disks. In the case `n = 1`, move this disk from peg `FROM` to peg `TO`.

2) Argument specification for the subprograms are given in the following tables.

Table 1 `TowerHanoiGame` arguments

Variable name	Input/Output	Meaning
<code>N</code>	Input	Number of disks
<code>FROM</code>	Input	The name of peg where <code>N</code> disks are mounted before running this subprogram
<code>INTERMEDIATE</code>	Input	The name of peg where disks could be moved to or from during running this subprogram
<code>TO</code>	Input	The name of peg where <code>N</code> disks are finally mounted after running this subprogram

Subquestion 2

From the answer group below, select the correct answers to be inserted into the blanks

through .

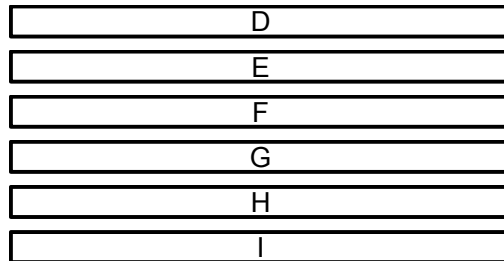
Note: Answers could be the same.

The solution for 3 disks is produced by the call

```
TowerHanoiGame (3, "FROM ", "INTERMEDIATE ", "TO");
```

The output is as below:

Move the top disk from peg FROM to peg TO



Answer group:

- a) Move the top disk from peg FROM to peg INTERMEDIATE
- b) Move the top disk from peg FROM to peg TO
- c) Move the top disk from peg INTERMEDIATE to peg FROM
- d) Move the top disk from peg INTERMEDIATE to peg TO
- e) Move the top disk from peg TO to peg FROM
- f) Move the top disk from peg TO to peg INTERMEDIATE

Subquestion 3

From the answer group below, select the correct answers to be inserted into the blanks

through .

The Towers of Hanoi game moves the disks 7 times for 3 disks ($n=3$).

This program moves the disks times for 4 disks, times for 5 disks, and times for n disks.

Answer group:

- a) 11
- b) 15
- c) 31
- d) 32
- e) $2n + 1$
- f) $2^n - 1$

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

[Program Description]

One program is developed to classify the students into two classes X and Y at a training center. Class X contains the more competent students and Class Y contains less competent students who didn't pass the tests that organized in the center. The center has the appropriate training plans for each class to get best result.

- 1) There are 80 students registered for the training.
- 2) Each test consists of 80 questions. The questions are classified by the topics. The number of questions and minimum number of correct answers (pass criteria) in each topic for each test is shown in the Table1.

Table1 Topic_Table (Number of questions for each topic and pass test criteria)

	0	1	← Column
Row ↓	Number of questions	Minimum number of correct answers	Topic's comment
0	6	4	Topic 1: Computer Science fundamentals
1	40	32	Topic 2: Computer system
2	10	6	Topic 3: System development and operation
3	6	4	Topic 4: Network Technology
4	6	4	Topic 5: Database Technology
5	6	4	Topic 6: Security and Standardization
6	6	4	Topic 7: Computerization and management

- 3) 80 questions are ordered by the topics as shown below:
 - Questions 1 to 6 belong to Topic 1
 - Questions 7 to 46 belong to Topic 2
 - Questions 47 to 56 belong to Topic 3
 - Questions 57 to 62 belong to Topic 4
 - Questions 63 to 68 belong to Topic 5
 - Questions 69 to 74 belong to Topic 6
 - Questions 75 to 80 belong to Topic 7
- 4) StudentID and test results of 80 questions (1: correct, 0: incorrect) for all 80 students are stored in the **Master_Table** as shown in the Table 2.

Table2 Master_Table

Row	StudentID														pass	Column
	0	1	2	3	4	5	6	...	j	..	77	78	79	80	81	
0	1	1	1	0	1	0	0	..		.	1	1	1	1		
1	2	1	1	1	0	0	1	1	1	0	1		
2	3	1	1	1	1	1	1	1	1	1	1		
3	4	1	1	1	0	1	0	0	1	1	1		
..																
i	i															
79	80	1	1	0	1	0	0	1	1	1	1		

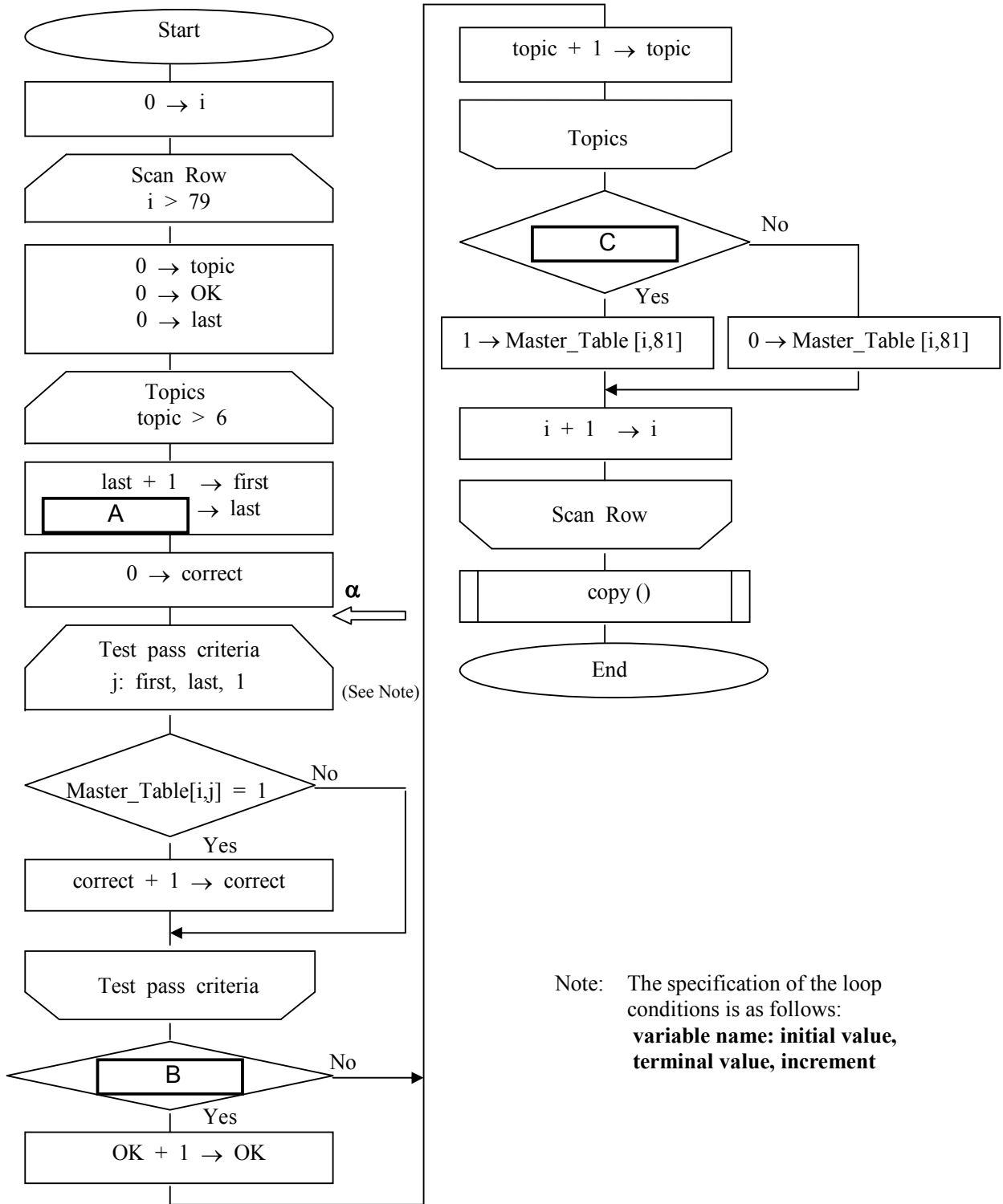
- 5) An element **Master_Table [i,j]**, where $(0 \leq i \leq 79, 1 \leq j \leq 80)$, corresponds to student “i” and the answer “j”.
- 6) For clarity, an example is given by extraction of the last row from the Table2. After a test, the test answers made by the student whose StudentID is 80 are looked like this:

Row	StudentID														pass
	0	1	2	3	4	5	6	...	j	...	77	78	79	80	81
79	80	1	1	0	1	0	0	1	1	1	1	

Question 1, 2, 4, ... : correct answer
 Question 3, 5, 6, ... : incorrect answer

- 7) After the test, the students that pass the test (is selected by the criteria, described in the Table1) are marked with pass = 1 in column **pass**, that means “correct answers” / “number of questions in the Topic” is greater than or equal to 4/6, 32/40, 6/10, 4/6, 4/6, 4/6, 4/6, corresponding to Topic 1 to Topic 7, respectively.
- 8) After the test, all the records of the passed students are copied to a table **class_X**, and rest of the records are copied to a table **class_Y**. This is done by the subroutine **copy()**.

[Flowchart]



Subquestion

From the answer groups below, select the correct answers to be inserted into the blanks through in the above flowchart.

And select the correct answer to be inserted into the blank in the following description.

After execution of the process indicated by the arrow α , when the variable **Topic** = 2, the correct combination of values for the variables **first** and **last** are .

Answer group for A:

- | | |
|--|--|
| a) $\text{first} + \text{Topic_Table}[\text{topic}, 0]$ | b) $\text{first} + \text{Topic_Table}[\text{topic}, 0] + 1$ |
| c) $\text{last} + \text{Topic_Table}[\text{topic}, 0]$ | d) $\text{last} + \text{Topic_Table}[\text{topic}, 0] + 1$ |

Answer group for B:

- | | |
|---|--|
| a) $\text{correct} > 0$ | b) $\text{correct} = 1$ |
| c) $\text{correct} \leq \text{Topic_Table}[\text{Topic}, 1]$ | d) $\text{correct} = \text{Topic_Table}[\text{Topic}, 1]$ |
| e) $\text{correct} \geq \text{Topic_Table}[\text{Topic}, 1]$ | |

Answer group for C:

- | | |
|-------------------------|-------------------------|
| a) $\text{correct} > 0$ | b) $\text{correct} = 1$ |
| c) $\text{correct} = 6$ | d) $\text{correct} = 7$ |
| e) $\text{OK} > 0$ | f) $\text{OK} = 1$ |
| g) $\text{OK} = 6$ | h) $\text{OK} = 7$ |

Answer group for D:

- | | |
|--|--|
| a) $\text{first} = 1, \text{last} = 6$ | b) $\text{first} = 7, \text{last} = 45$ |
| c) $\text{first} = 6, \text{last} = 46$ | d) $\text{first} = 7, \text{last} = 46$ |
| e) $\text{first} = 47, \text{last} = 56$ | f) $\text{first} = 57, \text{last} = 62$ |

6) The following function is available to the program:

Format: int replaceChar(unsigned char c)

Argument: c

Return value: If character c is a non-conversion character, 0 is returned.
Otherwise, 1 is returned.

[Program]

```
int replaceChar( unsigned char );

void URLEncode( unsigned char *input, unsigned char *output ) {

    const unsigned char chars[] = "0123456789ABCDEF";

    while( *input != '\0' ) {
        if ( replaceChar(  ) ) {
            *output++ = '%';
            *output++ = chars[  ];
            *output++ = chars[  ];
        } else {
            *output++ = *input;
        }
        ;
    }
    *output = '\0';
}
```

Subquestion

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

Answer group for A and D:

- | | | |
|------------|---------------|---------------|
| a) &input | b) (&input)++ | c) (*input)++ |
| d) **input | e) **input++ | f) *input |
| g) input | h) input++ | |

Answer group for B and C:

- | | |
|------------------|------------------|
| a) *input << 4 | b) *input >> 4 |
| c) *input & 0xF0 | d) *input & 0xF0 |
| e) *input ^ 0xF0 | f) *input ^ 0xF0 |
| g) *input 0xF0 | h) *input 0xF0 |

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

[Program Description]

This is a program that opens an input file containing the test results of all students in each grade in a particular institution and obtains the total scores in five subjects. It sorts the results in descending order of the total scores and writes them onto output files so that the overall ranking in each grade can be determined.

1) Input file `IN-FILE` is a sequential file which has the following record format, and stores the scores of all students in one grade. Here, a perfect score for each subject is 100 points.

Class number	Student number	Name	Japanese language	Mathematics	English language	Science	Social studies
2 digits	2 digits	20 digits	3 digits	3 digits	3 digits	3 digits	3 digits

2) Output file `OUT-FILE` is a sequential file which has the following record format, and stores data in descending order of the total scores.

Class number	Student number	Name	Total score
2 digits	2 digits	20 digits	3 digits

[Program]

(Line number)

```

1 DATA DIVISION.
2 FILE SECTION.
3 FD IN-FILE.
4 01 IN-REC          PIC X(39) .
5 FD OUT-FILE.
6 01 OUT-REC        PIC X(27) .
7 SD SORT-FILE.
8 01 SORT-REC.
9   02 CLASS-NO     PIC 9(2) .
10  02 STUDENT-NO    PIC 9(2) .
11  02 STUDENT-NAME PIC X(20) .
12  02 TOTAL        PIC 9(3) .
13 WORKING-STORAGE SECTION.
14 01 W-IN-REC.
15   02 STUDENT-ID.
16     03 CLASS-NO   PIC 9(2) .
17     03 STUDENT-NO PIC 9(2) .
18     03 STUDENT-NAME PIC X(20) .
19     02 SCORE      PIC 9(3) OCCURS 5 .

```

```

20 01 READ-STATUS PIC X(1) VALUE SPACE.
21     88 AT-END VALUE "E".
22 01 K PIC 9(1).
23 PROCEDURE DIVISION.
24 SORT-PROCEDURE.
25     SORT SORT-FILE DESCENDING KEY TOTAL
26     INPUT PROCEDURE IS READ-DATA
27     GIVING OUT-FILE.
28     STOP RUN.
29 READ-DATA.
30     OPEN INPUT IN-FILE.
31     PERFORM UNTIL AT-END
32     READ IN-FILE AT END
33     A
34     NOT AT END
35     MOVE IN-REC TO W-IN-REC
36     PERFORM RELEASE-DATA
37     END-READ
38     END-PERFORM.
39     CLOSE IN-FILE.
40 RELEASE-DATA.
41     B.
42     MOVE ZERO TO TOTAL.
43     PERFORM VARYING K FROM 1 BY 1 UNTIL K > 5
44     COMPUTE TOTAL = TOTAL + SCORE(K)
45     END-PERFORM.
46     RELEASE SORT-REC.

```

Subquestion 1

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

Answer group:

- a) INITIALIZE SORT-REC
- b) MOVE SPACE TO READ-STATUS
- c) MOVE SPACE TO SORT-REC
- d) MOVE STUDENT-ID TO SORT-REC
- e) PERFORM RELEASE-DATA
- f) SET AT-END TO TRUE

Subquestion 2

The program will be changed in such a way that the numbers of students and grade average points of all subjects will be displayed at the end of the program. From the answer group below, select the correct answers to be inserted into the blanks in the following table showing description of changes in the program.

Here, the number of students is assumed to be in the 1 to 9,999 range, and the average scores are truncated to 2 decimal places.

Action	Description of changes in program
To be added between line numbers 22 and 23.	01 STATISTICS. 02 STUDENT-TOTAL PIC 9(4) VALUE ZERO. 02 SUB-TOTAL PIC 9(8) VALUE ZERO OCCURS 5. 02 AVERAGE PIC 999.9.
To be added between line numbers 27 and 28.	DISPLAY "STUDENT:". DISPLAY STUDENT-TOTAL. DISPLAY "AVERAGE:". PERFORM VARYING K FROM 1 BY 1 UNTIL K > 5 COMPUTE AVERAGE = SUB-TOTAL(K) / STUDENT-TOTAL DISPLAY AVERAGE END-PERFORM.
To be added between line numbers 44 and 45.	<input type="text" value="C"/>
To be added after line number 46.	<input type="text" value="D"/>

Answer group:

- a) COMPUTE STUDENT-TOTAL = STUDENT-TOTAL + 1
- b) COMPUTE STUDENT-TOTAL = STUDENT-TOTAL + K
- c) COMPUTE SUB-TOTAL(K) = SUB-TOTAL(K) + SCORE(K)
- d) MOVE K TO STUDENT-TOTAL
- e) MOVE SCORE(K) TO SUB-TOTAL(K)
- f) MOVE TOTAL TO SUB-TOTAL(K)

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

[Program Description]

The program will read in numbers from the user, then search the number in the predefined sorted array.

In the program, a binary search algorithm is used for finding the number.

The binary search begins by comparing the number to the one in the middle of the array. If not matched, it is obvious whether the number would belong before or after that middle number, because the numbers in the array are sorted. The search then continues through the correct half in the same way.

[Program]

```
public class BinarySearch {
    private long[] a;

    private int numberOfElements;

    public BinarySearch(int max) {
        a = new long[max];
        numberOfElements = 0;
    }

    public int size() {
        return numberOfElements;
    }

    public int search(long searchKey) {
        return BSearch(searchKey, 0, numberOfElements - 1);
    }

    private int BSearch(long searchKey, int lowerBound, int upperBound) {
        int currentPosition;

        currentPosition = (lowerBound + upperBound) / 2;
        if (a[currentPosition] == searchKey)
            return currentPosition;
        else if (lowerBound > upperBound)
            return numberOfElements;
        else
        {
            if (A)
                return BSearch(searchKey, currentPosition + 1, upperBound);
            else
                return BSearch(searchKey, B, currentPosition - 1);
        }
    }
}
```

```

public void insert(long value) {
    a[numberOfElements] = value;
    numberOfElements++;
}

public static void main(String[] args) {
    int maxSize = 100;
    BinarySearch br = new BinarySearch(maxSize);

    br.insert(121);
    br.insert(130);
    br.insert(150);
    br.insert(226);
    br.insert(314);
    br.insert(369);
    br.insert(444);
    br.insert(527);
    br.insert(695);
    br.insert(719);
    br.insert(723);
    br.insert(808);
    br.insert(944);
    br.insert(1017);
    br.insert(1053);
    br.insert(1296);

    int searchKey = Integer.parseInt(args[0]);
    if ()
        System.out.println("Found " + searchKey);
    else
        System.out.println("Can't find " + searchKey);
}
}

```

Subquestion

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

Answer group for A:

- a) `a[currentPosition] < searchKey`
- b) `a[currentPosition] > searchKey`
- c) `a[currentPosition] == searchKey`
- d) `a[currentPosition+1] > searchKey`

Answer group for B:

- a) `lowerBound`
- b) `lowerBound + 1`
- c) `upperBound`
- d) `upperBound + 1`

Answer group for C:

- a) `br.search(searchKey) != br.size()`
- b) `br.search(searchKey) != br.size() - 1`
- c) `br.search(searchKey) == br.size()`
- d) `br.search(searchKey) == br.size() - 1`

Select one question from Q9 through Q11, mark (S) in the selection area on the answer sheet, and answer the question.
 If two or more questions are selected, only the first question will be graded.

Q9. Read the following description of a C program and the program itself, and then answer Subquestions 1 and 2.

[Program 1 Description]

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

1) Description on the notation of source programs

- (i) “Comments” handled by this program are character strings that start with “/*” and end with “*/”, excluding those included in character constants, character string literals, and comments.
- (ii) The type of usable characters is as follows.

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

(iii) Description as shown below is not used.

- Nested comments

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

- Three consecutive graphic characters

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

(iv) There are no grammatical errors.

- 2) Program 1 removes comments in accordance with the following procedure. Since the program simply processes the analyses of character constants, character string literals and comments, they may not be recognized correctly depending on the coding in source programs, resulting in a malfunction.
 - (i) When detecting a single quote or double quote, the program interprets it as the beginning of a character constant or character string literal, and then uses function `quote` to read and output the character string as it is until the program detects the corresponding single quote or double quote.
 - (ii) When detecting `“/*”`, the program interprets it as the beginning of a comment and skips characters before the first appearance of `“*/”`.
- 3) An execution example of the comment removal by Program 1 is shown below.

Input source program

```

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

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

```

Output results after the removal of comments

```

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

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

```

Fig. Execution Example of the Comment Removal

[Program 1]

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

main()
{
    int c1, c2;

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

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

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

Subquestion 1

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

Answer group:

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

[Program 2 Description]

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

- 1) The process to be implemented is divided into the three modes: character constant, character string literal, and comment.
- 2) An appearance of a single quote switches the “character constant mode” between ON and OFF. However, this does not apply to a piece of coding which is an expanded representation using a “\”, to a piece of coding within a character string literal, or to a piece of coding within a comment.
- 3) An appearance of double quotes switches the “character string literal mode” between ON and OFF. However, this does not apply to a piece of coding which is an expanded representation using a “\”, to a piece of coding within a character constant, or to a piece of coding within a comment.
- 4) An appearance of “/*” and “*/” switches the “comment mode” between ON and OFF. However, this does not apply to a piece of coding within a character constant or to a piece of coding within a character string literal.

[Program 2]

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

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

        if ( !c_mode ) { /* when comment mode is off          */
            /* '\ ' in character constant or character string literal? */
            if (  && c1 == '\\\ ' ) {
                putchar(c1);
                putchar(c2);
                c2 = getchar();
                continue;
            }
            /* single quote which is not inside a character string literal? */
            else if ( !quote2 && c1 == '\ ' )
                ;
            /* double quote which is not inside a character constant? */
            else if ( !quotel && c1 == '\" ' )
                ;
            /* '/' and '*' which is not inside a character constant
                and character string literal? */
            else if (  && c1 == '/' && c2 == '*' ) {
                ;
                c2 = getchar();
                continue;
            }
            putchar(c1);
        }

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

Subquestion 2

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

Answer group for A and D:

- | | |
|--------------------------------------|--|
| a) <code>!quote1</code> | b) <code>!quote2</code> |
| c) <code>(!quote1 !quote2)</code> | d) <code>(!quote1 && !quote2)</code> |
| e) <code>(quote1 quote2)</code> | f) <code>(quote1 && quote2)</code> |

Answer group for B, C and E:

- | | |
|----------------------------------|---|
| a) <code>c_mode = !c_mode</code> | b) <code>c_mode = quote1 && quote2</code> |
| c) <code>quote1 = !quote1</code> | d) <code>quote1 = !quote2</code> |
| e) <code>quote1 = quote2</code> | f) <code>quote2 = !quote1</code> |
| g) <code>quote2 = !quote2</code> | h) <code>quote2 = quote1</code> |

Q10. Read the following description of a COBOL program and the program itself, and then answer Subquestion.

[Program Description]

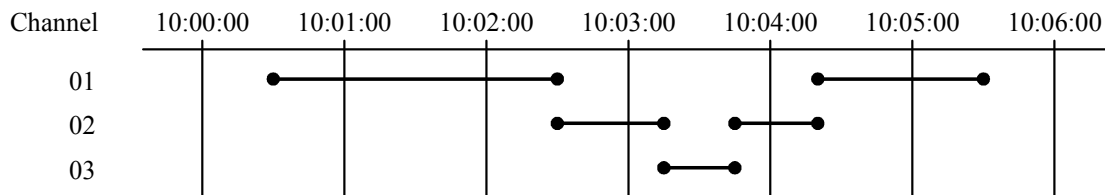
Five television channels can be received in a certain geographical area. This program reads the view data file that records TV programs viewed by surveyed households in that area on a given day, and the TV program data file that records the TV program data on the same day. It then calculates the average audience rating and prints it. The program rounds down the time in units of one minute. For instance, the time 10:00 is the figure obtained by rounding down the time between 10:00:00 and 10:00:59.

1) The view data file (VIEW-FILE) is a sequential file with the following record format:

Channel No. 2 digits	Detection start time		Detection end time	
	Hour 2 digits	Minute 2 digits	Hour 2 digits	Minute 2 digits

- (i) The view data file stores the audience data for a given day collected from the surveyed 600 sample households.
- (ii) Assume that there is one television set per household.
- (iii) Assume that the channel numbers range from 01 to 05.
- (iv) A record shows the channel viewed by a given household, indicating the “channel no.,” the “detection start time” and the “detection end time.”
- (v) Audience data is obtained by detecting the channels viewed by surveyed households at 00 second of every minute.

For instance, suppose that a certain household viewed TV from 10:00:30 to 10:05:30, during which it changed channels as follows:



Since channels are detected at 00 second of every minute, the data is recorded in the view file, rounded down in minute units as follows. Channel 03 was viewed but not recorded.

```
0110011002
0210031004
0110051005
```

- (vi) The range of the detection start time and the detection end time is from 00:00 to 23:59. No data extends to the second day.
 - (vii) The detection start time is equal to or less than the detection end time.
- 2) The program data file (PROGRAM-FILE) is a sequential file with the following record format:

Channel No. 2 digits	Program start time		Program end time		Program title 50 digits
	Hour 2 digits	Minute 2 digits	Hour 2 digits	Minute 2 digits	

- (i) The program data file stores program data for 5 channels, on the same day as in the view data file.
 - (ii) The range of the program start time and the program end time is from 00:00 to 23:59. No data extends to the second day.
 - (iii) The program start time is equal to or less than the program end time.
 - (iv) A record shows that a program is broadcast from the “program start time” to the “program end time.”
- 3) The program executes as follows:
- (i) The view count table (VIEW-COUNT-TABLE) is a two-dimensional table consisting of 5 (channels) × 1,440 (minutes). It summarizes the number of households by channel viewed in minute units and enters the data in each category of the table.
 - (ii) The average audience rating of a TV program is obtained using the following equation:

$$\begin{aligned} &\text{Average audience rating of program (\%)} \\ &= (\text{Sum of the time during which each household viewed that program}) \\ &\quad / (\text{Number of sample households} \times \text{broadcast time of the program}) \times 100 \end{aligned}$$

(iii) The channel, program start time, program end time, average audience rating (in percent, to the first decimal place), and the program name are printed in the order in which programs were read from the program data file, in the following format:

channel	from - to	rating(%)	program-title
02	2320 - 2359	8.3	Sports News
04	0430 - 0439	1.7	Good Morning
:	:	:	:

[Program]

```

DATA DIVISION.
FILE SECTION.
FD VIEW-FILE.
01 VIEW-REC.
   05 VIEW-CHANNEL          PIC 99.
   05 VIEW-START-HHMM.
     10 VIEW-START-HH      PIC 99.
     10 VIEW-START-MM      PIC 99.
   05 VIEW-END-HHMM.
     10 VIEW-END-HH        PIC 99.
     10 VIEW-END-MM        PIC 99.
FD PROGRAM-FILE.
01 PROGRAM-REC.
   05 PROG-CHANNEL          PIC 99.
   05 PROG-START-HHMM.
     10 PROG-START-HH      PIC 99.
     10 PROG-START-MM      PIC 99.
   05 PROG-END-HHMM.
     10 PROG-END-HH        PIC 99.
     10 PROG-END-MM        PIC 99.
   05 PROGRAM-TITLE        PIC X(50).
FD OUT-FILE.
01 OUT-REC                  PIC X(100).
WORKING-STORAGE SECTION.
01 SAMPLE-SIZE             PIC 9(4) VALUE 600.
01 M                       PIC 9(4).
01 END-OF-FILE             PIC X.
01 START-MMMM              PIC 9(4).
01 END-MMMM                PIC 9(4).
01 SUMMATION               PIC 9(9) BINARY.
01 PROG-RATING             PIC ZZ9.9.
01 VIEW-COUNT-TABLE.
   05 CHANNEL               OCCURS 5.
     10 COUNT-OF-MINUTE    OCCURS 1440 PIC 9(4) BINARY.

```

```

01 O-DATA.
  05 FILLER                PIC X(3) VALUE SPACE.
  05 O-PROG-CHANNEL        PIC 99.
  05 FILLER                PIC X(3) VALUE SPACE.
  05 O-PROG-START-HHMM.
    10 O-PROG-START-HH     PIC 99.
    10 O-PROG-START-MM     PIC 99.
  05 FILLER                PIC X(3) VALUE " - ".
  05 O-PROG-END-HHMM.
    10 O-PROG-END-HH       PIC 99.
    10 O-PROG-END-MM       PIC 99.
  05 FILLER                PIC X(2) VALUE SPACE.
  05 O-PROG-RATING         PIC ZZ9.9.
  05 FILLER                PIC X(6) VALUE SPACE.
  05 O-PROGRAM-TITLE       PIC X(50).
01 TITLE-LINE              PIC X(100)
    VALUE "channel from - to rating(%) program-title".
PROCEDURE DIVISION.
MAIN-PARAGRAPH.
  PERFORM EXPAND-VIEW-COUNT.
  PERFORM CALCULATE-RATING-AND-PRINT.
  STOP RUN.
EXPAND-VIEW-COUNT.
  A.
  OPEN INPUT VIEW-FILE.
  MOVE "N" TO END-OF-FILE.
  PERFORM UNTIL END-OF-FILE = "Y"
    READ VIEW-FILE AT END
    MOVE "Y" TO END-OF-FILE
  NOT AT END
    PERFORM SET-VIEW-COUNT
  END-READ
END-PERFORM.
CLOSE VIEW-FILE.
CALCULATE-RATING-AND-PRINT.
  OPEN INPUT PROGRAM-FILE OUTPUT OUT-FILE.
  WRITE OUT-REC FROM TITLE-LINE AFTER ADVANCING 1.
  B.
  PERFORM UNTIL END-OF-FILE = "Y"
    READ PROGRAM-FILE AT END
    MOVE "Y" TO END-OF-FILE
  NOT AT END
    PERFORM CALCULATE-RATING
    MOVE PROG-CHANNEL      TO O-PROG-CHANNEL
    MOVE PROG-START-HHMM  TO O-PROG-START-HHMM
    MOVE PROG-END-HHMM    TO O-PROG-END-HHMM
    MOVE PROG-RATING      TO O-PROG-RATING
    MOVE PROGRAM-TITLE    TO O-PROGRAM-TITLE
    WRITE OUT-REC FROM O-DATA AFTER ADVANCING 1
  END-READ
END-PERFORM.
CLOSE PROGRAM-FILE OUT-FILE.

```

```

SET-VIEW-COUNT.
  COMPUTE START-MMMM = VIEW-START-HH * 60 + VIEW-START-MM + 1.
  COMPUTE END-MMMM = VIEW-END-HH * 60 + VIEW-END-MM + 1.
  PERFORM VARYING M FROM START-MMMM BY 1 UNTIL M > END-MMMM
    
  END-PERFORM.
CALCULATE-RATING.
  COMPUTE START-MMMM = PROG-START-HH * 60 + PROG-START-MM + 1.
  COMPUTE END-MMMM = PROG-END-HH * 60 + PROG-END-MM + 1.
  .
  PERFORM VARYING M FROM START-MMMM BY 1 UNTIL M > END-MMMM
    
  END-PERFORM.
  COMPUTE PROG-RATING ROUNDED
    = .

```

Subquestion

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

Answer group for A, B and D:

- a) COMPUTE SUMMATION = END-MMMM - START-MMMM
- b) INITIALIZE VIEW-COUNT-TABLE
- c) MOVE "N" TO END-OF-FILE
- d) MOVE "Y" TO END-OF-FILE
- e) MOVE SAMPLE-SIZE TO SUMMATION
- f) MOVE ZERO TO SUMMATION
- g) MOVE ZERO TO VIEW-COUNT-TABLE

Answer group for C and E:

- a) ADD 1 TO COUNT-OF-MINUTE (VIEW-CHANNEL M)
- b) ADD 1 TO COUNT-OF-MINUTE (VIEW-CHANNEL M + 1)
- c) ADD 1 TO SUMMATION
- d) ADD COUNT-OF-MINUTE (PROG-CHANNEL M) TO SUMMATION
- e) COMPUTE SUMMATION = COUNT-OF-MINUTE (PROG-CHANNEL M + 1)
- f) COMPUTE SUMMATION = COUNT-OF-MINUTE (VIEW-CHANNEL M) + 1
- g) MOVE 1 TO COUNT-OF-MINUTE (VIEW-CHANNEL M)

Answer group for F:

- a) $100 * \text{SUMMATION} / ((\text{END-MMMM} - \text{START-MMMM}) * (\text{SAMPLE-SIZE} + 1))$
- b) $100 * \text{SUMMATION} / ((\text{END-MMMM} - \text{START-MMMM} + 1) * \text{SAMPLE-SIZE})$
- c) $100 * \text{SUMMATION} / (\text{END-MMMM} - \text{START-MMMM}) / \text{SAMPLE-SIZE}$
- d) $100 * \text{SUMMATION} / (\text{END-MMMM} - \text{START-MMMM} + 1) / (\text{SAMPLE-SIZE} + 1)$

Q11. Read the following description of a Java program and the program itself, and then answer Subquestions 1 and 2.

[Program Description]

This program simulates the processing by an automated ticket gate installed at each station of a railroad line comprising four stations A, B, C, and D, with A serving as the starting point and D as the terminal point. On this line, the following two types of fare certificates can be used: one-way ticket (hereinafter, “ticket”); and prepaid card (hereinafter, “card”).

A fare on the line is decided based on the distance between stations. The fare for up to 4 kilometers is 120 yen (this fare is called the base fare). An amount of 30 yen is added for each additional 2 kilometers. Any additional distance less than 2 kilometers is rounded up to 2 kilometers. For example, if the distance is 7 kilometers, the fare is 180 yen.

The station of embarkation is recorded on a ticket when a passenger passes through the automated ticket gate in that station to enter the platform area. When he/she passes through the automated ticket gate to leave the platform area in the station of disembarkation, the fare is calculated, and if the amount paid for the ticket is insufficient, the gate is closed to prevent him/her from leaving the platform area. Once a ticket is used, it becomes invalid. On this line, a passenger can enter the platform area through the automated ticket gate in any station, thus being able to embark, regardless of the station that issued the ticket. For instance, with a ticket issued at Station A, a passenger can enter the platform area through the automated ticket gate in Station B.

The station of embarkation is recorded on a card when a passenger enters the platform area through the automated ticket gate in that station. At this time, if the balance on the card is zero, the gate is closed to prevent him/her from entering the platform area. When he/she leaves the platform area through the automated ticket gate in the station of disembarkation, a fare adjustment is processed. Namely, the fare is calculated and is subtracted from the balance on the card. At this time, if this balance is less than the amount of the fare, the gate is closed to prevent him/her from leaving the platform area.

Class `Line` represents the railroad line. Method `getFare` calculates the fare on the basis of a given distance, and returns this information.

Class `Gate` signifies an automated ticket gate. Fields `A`, `B`, `C` and `D` in class `Line` are instances of `Gate`, and represent the automated ticket gates installed at stations A, B, C and D, respectively. The constructor and each method perform the following processing.

- 1) The constructor generates an instance of `Gate`. A station name is specified as the first argument, and the distance from station A, which is the starting point of the line,

is specified as the second argument.

- 2) Method `enter` performs processing when a passenger enters the platform area through an automated ticket gate. If the fare certificate is not valid, the gate is closed. If entrance processing is carried out normally, information on the station of embarkation is recorded on this fare certificate.
- 3) Method `exit` performs processing when a passenger leaves the platform area through an automated ticket gate. If the fare certificate is invalid in that, for example, the amount (balance) on the fare certificate is insufficient, then the gate is closed.
- 4) Methods `open` and `close` output messages for opening and closing a gate, respectively.

Abstract class `Ticket`, which represents a fare certificate for this line, is inherited in defining a ticket and a card. The constructor and each method perform the following processing.

- 1) The constructor retains a purchase amount on the fare certificate as the initial value.
- 2) Method `getValue` returns the fare certificate amount (balance) as it is at the point in time when it is called.
- 3) Method `adjustValue` performs fare adjustment processing if necessary.
- 4) Method `deduct` deducts the amount specified as an argument from the amount (balance) of the fare certificate, and updates the amount (balance).
- 5) Method `setOrigin` records a specified Gate as the station of embarkation. If `null` is specified, the record of the station of embarkation is deleted.
- 6) Method `getOrigin` returns the station of embarkation that is recorded. If this station is not recorded, `null` is returned.

Class `OneWayTicket` represents a ticket, and class `PrepaidCard` represents a card. In the processing of either fare certificate, an abstract method is implemented, and the method in class `Ticket` is overridden, as necessary.

[Program 1]

```
public final class Line {
    public static final Gate A = new Gate("A", 0);
    public static final Gate B = new Gate("B", 5);
    public static final Gate C = new Gate("C", 8);
    public static final Gate D = new Gate("D", 14);

    public static int getFare(int distance) {
        return 120 + (Math.max(distance - 3, 0) / 2) * 30;
    }
}
```

[Program 2]

```
public class Gate {
    private final String name;
    private final int distance;

    public Gate(String name, int distance) {
        this.name = name;
        this.distance = distance;
    }

    public void enter(Ticket ticket) {
        if (ticket.isValid() && ticket.getOrigin() == null) {
            A;
            open();
        } else {
            close();
        }
    }

    public void exit(Ticket ticket) {
        Gate origin = ticket.getOrigin();
        if (origin != null) {
            int d = Math.abs(origin.distance - distance);
            int fare = Line.getFare(d);
            if (B) {
                ticket.adjustValue(fare);
                ticket.setOrigin(null);
                open();
                return;
            }
        }
        close();
    }

    private void open() { System.out.println(name + ": open"); }
    private void close() {
        System.out.println(name + ": closed");
    }
}
```

[Program 3]

```
public abstract class Ticket {
    private Gate origin;
    private int value;

    public Ticket(int value) {
        this.value = value;
    }
}
```

```

public int getValue() { return value; }

public void deduct(int amount) { value -= amount; }

public void setOrigin(Gate gate) { origin = gate; }

public Gate getOrigin() { return origin; }

public abstract void adjustValue(int amount);

public abstract boolean isValid();
}

```

[Program 4]

```

public class OneWayTicket extends Ticket {
    private boolean valid = true;

    public OneWayTicket(int value) {
        C;
    }

    public void setOrigin(Gate gate) {
        super.setOrigin(gate);
        if (gate == null)
            valid = false;
    }

    public void adjustValue(int amount) { }

    public boolean isValid() { return valid; }
}

```

[Program 5]

```

public class PrepaidCard extends Ticket {
    public PrepaidCard(int value) {
        C;
    }

    public void adjustValue(int amount) { deduct(amount); }

    public boolean isValid() {
        return getValue() > 0;
    }
}

```

Subquestion 1

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

Answer group for A:

- a) `ticket.setOrigin(Line.A)`
- b) `ticket.setOrigin(Line.D)`
- c) `ticket.setOrigin(null)`
- d) `ticket.setOrigin(this)`
- e) `ticket.setOrigin(ticket)`

Answer group for B:

- a) `ticket.getValue() < fare`
- b) `ticket.getValue() <= fare`
- c) `ticket.getValue() == fare`
- d) `ticket.getValue() > fare`
- e) `ticket.getValue() >= fare`

Answer group for C:

- a) `super()`
- b) `super(this)`
- c) `super(value)`
- d) `super(); this.value = value`
- e) `this(value)`

Subquestion 2

It was decided that a new type of fare certificate be sold. A fare certificate of this type permits a passenger to freely embark or disembark at all stations within 24 hours from the time that the fare certificate is issued. Twenty-four hours after such a fare certificate is issued, the passenger can leave the platform area in any station but cannot enter any such area. For this, it is desired that a new class that inherits abstract class `Ticket` be defined to permit the use of this type of certificate without making any modification to class `Gate`. Processing by the constructor and methods of this class are shown in the following table. From the answer group below, select the correct answers to be inserted into the blanks in the table. Here, `value` in the answer group is a field `value` of class `Ticket`, and it is assumed that the initial value is set by the constructor and that the value is obtained by method `getValue`.

Constructor and methods	Processing
Constructor	D
Method <code>getValue</code>	E
Method <code>setOrigin</code>	The same as is defined in the superclass
Method <code>getOrigin</code>	The same as is defined in the superclass
Method <code>adjustValue</code>	No processing is performed (methods proper do not contain any statements).
Method <code>isValid</code>	F

Answer group:

- a) 0 is returned at all times.
- b) An amount that is theoretically too large to be spent in 24 hours is set as the initial value of `value`.
- c) Method `deduct` is called, with `amount` as an argument.
- d) No processing is performed (methods proper do not contain any statements).
- e) The base fare on the line is returned at all times.
- f) The highest fare on the line (the maximum value among all fares) is set as the initial value of `value`, and the fare certificate issuance time is recorded.
- g) The same as is defined in the superclass.
- h) `true` is returned only when the call time is within 24 hours from the fare certificate issuance time stored in the instance. In other cases, `false` is returned.
- i) `true` is returned only when `value` is not less than the base fare on the line. In other cases, `false` is returned.
- j) `true` is returned only when `value` is not less than the highest fare on the line (the maximum value among all fares). In other cases, `false` is returned.