October 2008

Fundamental IT Engineer Examination (Afternoon)

Questions must be answered in accordance with the following:

Question Nos.	Q1 – Q5	Q6 , Q7	Q8 , Q9
Question Selection	Compulsory	Select 1 of 2	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 (Q6-Q7 and Q8-Q9)**

   Mark the Ⓢo 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

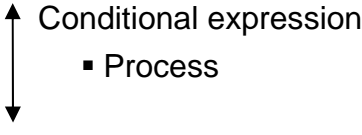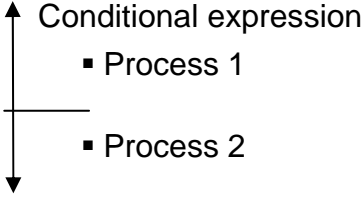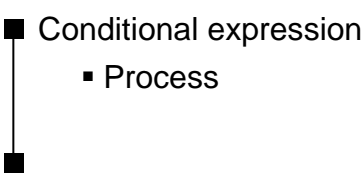   Since the correct answer is "b)" (October), mark your answer sheet as follows:

   [Sample Answer]

1	Ⓐ	●	Ⓒ	Ⓓ

[Explanation of the Pseudo-Code Description Format]

Pseudo-Language Syntax	Description
○	Declares names, types, etc. of procedures, variables, etc.
▪ Variable ← Expression	Assigns the value of an Expression to a Variable.
▲ Conditional expression ▪ Process	A selection process. If the Conditional expression is True, then Process is executed.

▲ 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. Process is executed while the Conditional expression is True.

[Operator]

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

[Logical type constant]

```
true  false
```

Q1 Read the following description of a program for the naive string-matching and the program itself, and then answer the Subquestions 1 and 2.

Assume that a text of length **n** is stored in an array T[1], T[2], ..., T[n] and that the pattern of length **m** is stored in an array P[1], P[2], ..., P[m]. It is said that the pattern occurs with shift **s** in text **T** (or, equivalently, that pattern **P** occurs beginning at position **s+1** in text **T**), if $0 \le s \le n\text{-}m$ and T[s+i]=P[i] (for i = 1, 2, …, **m**). If **P** occurs with shift **s** in **T**, then **s** is called a valid shift; otherwise, **s** is called an invalid shift. The string-matching algorithm is to find all valid shifts with which a given pattern **P** occurs in a given text **T** and number **k** of all occurrences. A pseudo code for the naive string-matching algorithm is shown below.

```
1    n ← length(T)
2    m ← length(P)
3    k ← 0
4    for  s = 0, 1, …,  [   A   ]
5         f ← 1
6         for  i = 1, 2, …, m
7              if  T[s+i] ≠ P[i]
8                   then  f ← 0
9         if  [   B   ]
10             then   k ←  [   C   ]
11                    print  "Pattern occurs with shift", s
12   print k
```

Subquestion 1
From the answer groups below, select the correct answers to be inserted in the blanks [   ] in the above program.

Answer group for A
   a) m
   b) n
   c) n-m
   d) n-m+1

Answer group for B

    a) $f = 0$

    b) $f = 1$

    c) $s = 0$

    d) $s = 1$

Answer group for C

    a) 1

    b) $f+1$

    c) $k+1$

    d) $s+1$

Subquestion 2

From the answer group below, select the correct answer for valid shift(s) and number of occurences of the pattern P=**abaa** in the text T=**abtabaabdabaa.**

Answer group

a)	s=3	k=1	
b)	s=4	k=1	
c)	s=3	s=9	k=2
d)	s=4	s=10	k=2

Q2 Read the following description concerning the processing of a school's schedule, and then answer Subquestions 1 through 3.

The registrar maintains a database consisting of tables named as Room, Course, Teacher and Class. The Room table contains all the rooms of the school. The Course table contains the courses offered. Identical courses are distinguished by the section id. One teacher and one class are assigned for each course and section. Students are grouped into classes and each class is scheduled as a block. Thus, all students of a class will have the same schedule and set of courses. The Class table contains the classes and students of that class.

Room

Room_id	Room_loc	Room_size

Course

Course_id	Section_id	Class_id	Teacher_id	Course_name	Units

Teacher

Teacher_id	Teacher_name

Class

Class_id	Student_id	Student_name

Subquestion 1

From the answer group below, select which is the most appropriate combination of fields that should compose the PRIMARY KEY of Course table considering that it is in the 3rd Normal Form?

Answer group
a) Course_id
b) Course_id, Section_id
c) Course_id, Section_id, Class_id
d) Course_id, Section_id, Class_id, Teacher_id

Subquestion 2

The registrar creates a Schedule table as shown below. The Schedule table contains the day (coded as 0: Sunday, 1: Monday, …, 6: Saturday), start time, end time, the class, the course, the section and the room.

Schedule (sample entries)

Day	Start_time	End_time	Class_id	Course_id	Section_id	Room_id
1	0700	0800	1-A	ECON	A	301
3	0700	0800	1-A	ECON	A	301
5	0800	0900	1-A	ECON	A	302
2	0800	0900	1-A	PHYSICS	C	204
4	0800	0900	1-A	PHYSICS	C	204
5	0900	1000	1-A	PHYSICS	C	204

```
CREATE  VIEW  Teacher_Sched  (TS_name,  TS_day,  TS_start,  TS_end,
                TS_course_name, TS_section)
        AS SELECT [      A      ]
        FROM Schedule_Table ST, Course_Table CT, Teacher_Table TT
        WHERE [        B        ]
        ORDER BY TT.Teacher_name, ST.Day, ST.Start_time
```

A schedule for each teacher is then presented and inspected for possible conflicts with the teacher's personal schedules. From the answer group below, select the correct answers to be inserted in the blanks [   ] in the above SQL statement.

Answer group

a) `TT.Teacher_name,  ST.Day, ST.Start_time, ST.End_time,`
   `CT.Course_name, CT.Seciton_id`

b) `TT.Teacher_name, ST.Day, ST.Start_time, ST.End_time,`
   `CT.Course_name, ST.Section_id`

c) `ST.Course_id = CT.Course_id AND ST.Room_id = RT.Room_id`

d) `ST.Course_id = CT.Course_id AND CT.Teacher_id = TT.Teacher_id`

e) `ST.Course_id = CT.Course_id AND ST.Section_id = CT.Section_id`
   `AND CT.Teacher_id = TT.Teacher_id`

f) `ST.Section_id = CT.Section_id AND CT.Teacher_id = TT.Teacher_id`

Subquestion 3

The Course table is as follows:

Course

Course_id	Section_id	Class_id	Teacher_id	Course_name	Units
ECON	A	1-A	F-0145	Economics	2
PHYSICS	B	1-A	F-0250	Physics	4
MAT-01	D	2-C	F-0201	Algebra	3
WHIST	E	1-A	F-0125	World History	2
ECON	B	1-A	F-0145	Economics	2
BIO	A	2-B	F-0250	Biology	4
MAT-01	E	2-B	F-0201	Algebra	3
CHEM	A	2-C	F-0250	Chemistry	4
ECON	C	2-B	F-0145	Economics	2
PHYSICS	C	2-C	F-0250	Physics	4
MAT-02	A	2-B	F-0201	Trigonometry	3
WHIST	F	2-B	F-0125	World History	2

From the answer group below, select the result of the following SQL statement.

```
SELECT Teacher_id, SUM(Units)
    FROM Course_Table
    WHERE Class_id <> '1-A'
    GROUP BY Teacher_id
    HAVING COUNT(Teacher_id) > 2
```

Answer group

a)

F-0145	6
F-0201	9
F-0250	16

b)

F-0125	2
F-0145	2
F-0201	9
F-0250	12

c)

F-0201	9
F-0250	12

d)

F-0201	9
F-0250	16

Q3 Read the following description concerning the flow control of a network, and then answer Subquestions 1 through 5.

In packet-based computer networks like the Internet, flow control is an important technique which is used to avoid data overrun at the receiver in the process of data exchange in both the data link layer (data frame exchange) and the transport layer (data packet exchange).

The figure below shows the simplest flow control protocol, called "stop-and-wait" flow control, as used in a data link layer. In stop-and-wait flow control, the frames will be sent to the receiver in sequence. Each frame sent out by the sender needs an explicit reply frame, called "acknowledgement (ACK)", being sent back from the receiver before the next packet can be sent out by the sender.

One performance metric for flow control protocols is efficiency, which in this question is defined as:

      Efficiency = Time to transmit user's data / Time for total transmission

Assume that the following parameters are used in the flow control protocols:
- There is only one hop between the sender and the receiver.
- Each data FRAME contains 10,000 bits of user's data and 100 bits of protocol header.
- The ACK (acknowledgement) is 100 bits in size.
- The link data rate between the sender and the receiver is 1Mbit/s.
- The distance between the sender and the receiver is 6×10^6 m (e.g. a low orbit satellite)
- The speed of each data bit propagating over the link is 3×10^8 m/s.
- The time to transmit data consists of only the *transmission delay* and the *propagation delay*. (Processing and queuing delays are negligible, that is, 0.)
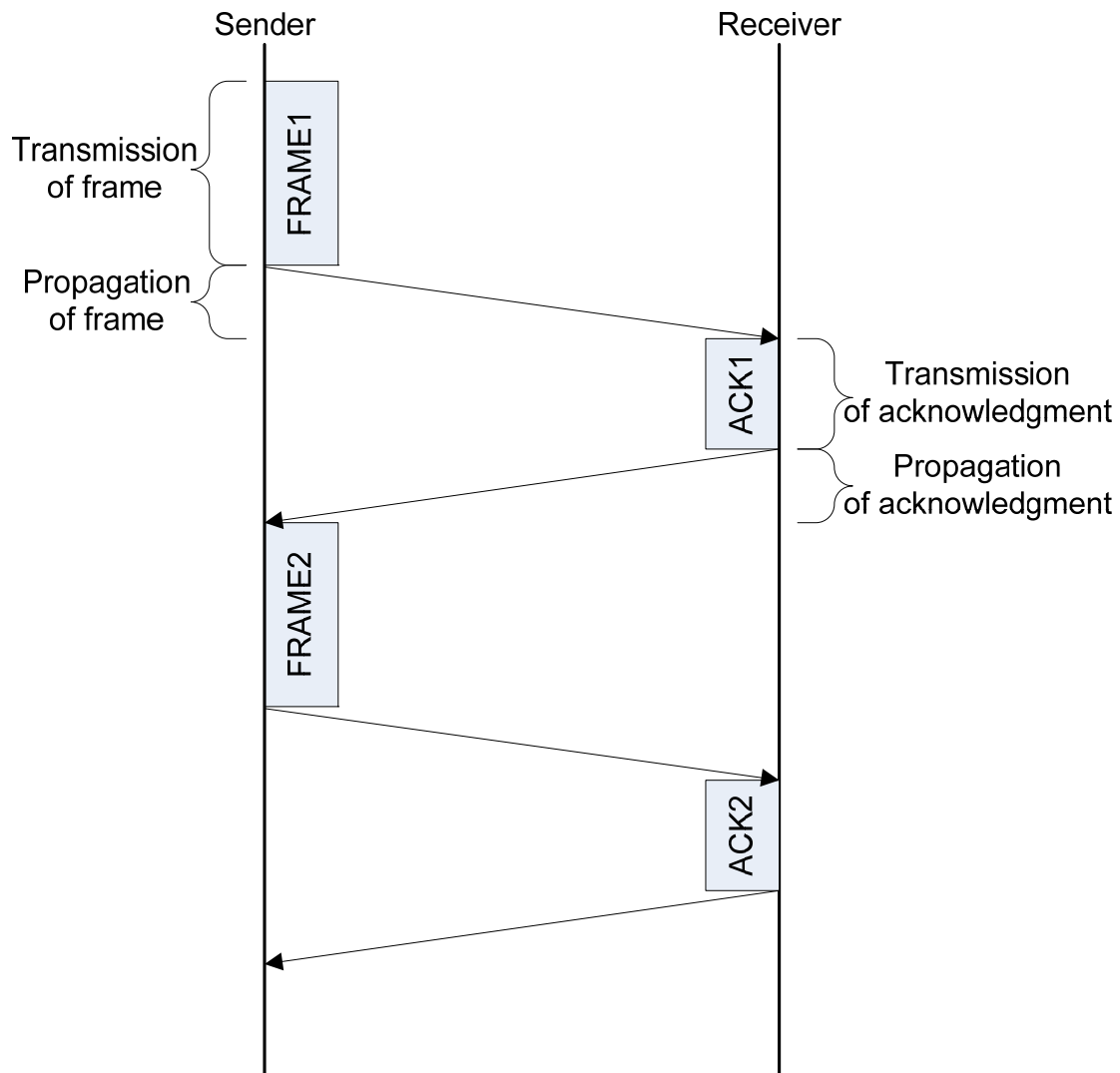
- 6 -

Figure   The model of stop-and-wait flow control

Subquestion 1

What is the transmission time (in milliseconds) of the single data frame (including header)?

Answer group

      a) 1.0   b) 1.1   c) 1.2   d) 10.0

      e) 10.1   f) 10.2   g) 20.0   h) 20.1

      i) 20.2

Subquestion 2

What is the propagation delay (in milliseconds) for the acknowledgement?

Answer group

    a) 1.0   b) 1.1   c) 1.2   d) 10.0

    e) 10.1   f) 10.2   g) 20.0   h) 20.1

    i) 20.2

Subquestion 3

What is the time (in milliseconds) for the total transmission when using the stop-and-wait flow control protocol (that is, the time between starting to send the first frame and receiving the entire ACK for that first frame)?

Answer group

    a) 21.0   b) 21.1   c) 21.2   d) 40.1

    e) 40.2   f) 40.3   g) 50.0   h) 50.1

    i) 50.2

Subquestion 4

What is the maximum efficiency of the stop-and-wait flow control protocol?

Answer group

    a) 0.11   b) 0.199   c) 0.22   d) 0.299

    e) 0.33   f) 0.399   g) 0.44   h) 0.499

Another popular flow control protocol is the "Sliding Window" flow control protocol. The Sliding Window flow control protocol is generally more efficient than the stop-and-wait flow control protocol. The major difference between the two protocols is the number of frames which is allowed to be sent from the sender without waiting for the explicit acknowledgement (ACK). In Sliding Window, up to W frames can be sent without an ACK being received. Assume the receiver only sends an ACK after receiving W frames. The single ACK implicitly acknowledges receipt of all W frames. Consider the Sliding Window flow control protocol where the window size (W) is set to 5 and the same protocol parameters as described above are used.

Subquestion 5

When sending 5 frames (for the full window size), what is the maximum efficiency of the Sliding Window flow control protocol?

Answer group
    a) 0.11   b) 0.22   c) 0.33   d) 0.44
    e) 0.55   f) 0.66   g) 0.77   h) 0.88

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

[Program Description]

The MergeSort is a subprogram to sort integer values that are stored in an array in ascending order.

1) The Num items of integers (Num>=2) to be sorted are stored in an array of global variables A[0], A[1], …, A[Num-1].
2) The MergeSort uses a recursive solution.
3) The procedure for sorting is as follows.
   (i)   Divide array into two pieces of left and right by middle element.
   (ii) Sort the left and right halves.
  (iii) Merge the two sorted halves into one sorted array comparing successive pairs of element, one from each half, moving the smaller of each pair to the "final" array.
4) Temporary array TempArray is used to merge two halves.
5) Fig. 1 shows the contents of the array A (given and final).

Array A (Given)

9	6	5	2	15	13	4	3	0

Array A (Final)

0	2	3	4	5	6	9	13	15

Fig. 1 Sorting Array A Using MergeSort

6) The specifications of the subprogram arguments are shown in Table 1 and Table 2.

Table 1 Specifications of the Argument for MergeSort

Argument Name	Data Type	Input/ Output	Meaning
A	Integer type	Input	Array to sort
First	Integer type	Input	Index of the first element of the array A
Last	Integer type	Input	Index of the last element of the array A

Table 2 Specifications of the Argument for Merge

Argument Name	Data Type	Input/Output	Meaning
A	Integer type	Input	Array to sort
LeftFirst	Integer type	Input	Index of the first element of the subarray in left half
LeftLast	Integer type	Input	Index of the last element of the subarray in left half
RightFirst	Integer type	Input	Index of the first element of the subarray in right half
RightLast	Integer type	Input	Index of the last element of the subarray in right half

[Program]

```
o Integer type : A[1000000]
o MergeSort(Integer type: A,
          Integer type: First, Integer type: Last)
o Integer type: Middle
  First<Last
            Middle ← (First+Last)/2
            MergeSort(A,First,Middle)
            MergeSort(A,Middle+1,Last)
            Merge(A, First, Middle, Middle+1, Last)


                    /* Merge the two sorted halves together*/
o Merge(Integer type: A,
       Integer type: LeftFirst, Integer type: LeftLast,
       Integer type: RightFirst, Integer type: RightLast)
```

o Integer type TempArray[1000000]

o Integer type: Index, SaveFirst

Index ← LeftFirst

SaveFirst ← LeftFirst

■ | A | and | B |

  ↑ | C |

        TempArray[Index] ← A[LeftFirst]

        LeftFirst ← LeftFirst+1

        _____

        TempArray[Index] ← A[RightFirst]

        RightFirst ← RightFirst+1

  ↓

 Index ← Index+1

■
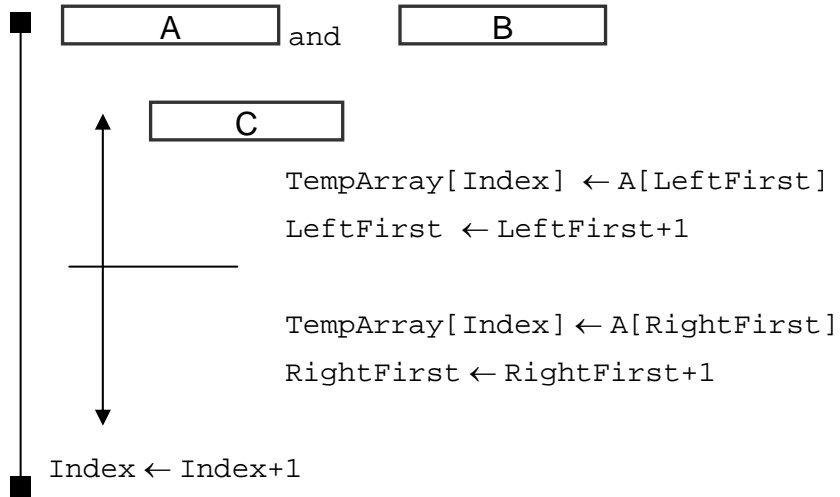
/* Copy any remaining elements from left half to TempArray. */

■ LeftFirst <= LeftLast

     TempArray[Index] ← A[LeftFirst]

   LeftFirst ← LeftFirst+1

   Index ← Index+1

■

/* Copy any remaining elements from right half to TempArray. */

■ RightFirst<=RightLast

     TempArray[Index] ← A[RightFirst]

   RightFirst ← RightFirst+1

   Index ← Index+1

■

/* Copy sorted elements from TempArray back into array A. */

■ Index: SaveFirst, Index<=RightLast, +1

     A[Index]=TempArray[Index]

■

Subquestion 1

From the answer groups below, select the correct answers to be inserted in the blanks

[   ] in the above program.

Answer group for A

a) `LeftFirst <= LeftLast`   b) `LeftFirst <= RightFirst`

c) `First < Last`   d) `LeftFirst < RightFirst`

Answer group for B

a) `LeftFirst >= RightFirst`   b) `LeftFirst > RightFirst`

c) `First < Last`   d) `RightFirst <= RightLast`

Answer group for C

a) `A[LeftFirst] < A[RightFirst]`   b) `A[LeftFirst] < A[LeftLast]`

c) `A[First] < A[Last]`   d) `A[RightFirst] <= A[RightLast]`

Subquestion 2

Using the array A (Given) in the Fig. 1, which of the following values is the element with the RightFirst index of array A(Given).

Answer group

a) 9   b) 13   c) 6   d) 15

Subquestion 3

Assume that executing the program using the array A (Given) in the fig. 1. When "Merge" is called at the 2nd time, what are the values of LeftFirst, LeftLast, RightFirst and RightLast?

Answer group

	LeftFirst	LeftLast	RightFirst	RightLast
a)	0	1	2	2
b)	0	2	3	4
c)	5	5	6	6
d)	5	6	7	8

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

A program is developed to score answer sheets as described in [Program Design Description] below, and then to grade examinees' answer results by calculating Correct Response Rate.

[Program Design Description]
 (1) Questions must be answered in accordance with the following:

Question Nos	Q1 – Q5	Q6 - Q9	Q10 - Q13
Question Selection	Compulsory	Select 1 of 4	Select 1 of 4

(2) The following is the extract of Answer Sheet:
The examinees' Answer Sheets, sheet by sheet, are passed through OMR equipment for recognition of solid rounded marks of answer options chosen by the examinees.

For the questions Q6 through Q9 and Q10 through Q13, questions selected to answer are marked by ⑤corresponding to item (1) above.
The following example shows that the questions Q1 through Q5 are compulsory, and the Q7 and Q10 are chosen (marked in the Answer Sheet) by an examinee.

Selection Column (Mark your selections.)

Q 1   ●   Q 6   ⑤   Q 10   ⑤

Q 2   ●   Q 7   ⑤   Q 11   ⑤

Q 3   ●   Q 8   ⑤   Q 12   ⑤

Q 4   ●   Q 9   ⑤   Q 13   ⑤

Q 5   ●

The following extract of Answer Sheet shows the answers for each question by an examinee:
answer a for A1, answer d for A2, answer c for A3, and answer b for A4.

Answer No.	Question	Sub-Question	Branch- Question	Answers					
A1	1	1	A	ⓐ	ⓑ	ⓒ	ⓓ	ⓔ	ⓕ
A2	1	1	B	ⓐ	ⓑ	ⓒ	ⓓ	ⓔ	ⓕ
A3	1	2		ⓐ	ⓑ	ⓒ	ⓓ		
A4	2	1		ⓐ	ⓑ	ⓒ	ⓓ		

In the above example, examinee identification information, represented in the form of marks, such as Examinee Number2, Date of Birth, is not shown.

(3) **Results** table:

The OMR Output Results table (hereafter, Results table) is looked as shown below. Here, N denotes total number of examinees, and A denotes total number of branch-questions

Result

	1	2	3	4	5	6	7	8	...	A+5
	ExamineeNo2	BirthDt	SelectionColumn1	SelectionColoumn2	SelectionColumn3	A1	A2	A3	⋮	⋮
1	0001	19831219	11111	0010	0010	a	c	c	…	e
2	0004	19840405	11111	1000	1000	a	d	b	…	e
3	0006	19820401	11111	0010	0010	a	d	c	…	d
…	…	…	…	…	…	…	…	…	…	…
5	0012	19821221	11111	1000	1000	b	d	c	…	e
6	0013	19841029	11111	0010	0010	a	e	c	…	e
7	0014	19840604	11111	0010	0010	a	d	d	…	e
…	…	…	…	…	…	…	…	…	…	…
N	0016	19851129	11111	0010	0010	a	d	c	a	f

The fields A1, A2, A3,… are the answers of an examinee in test examination, in the consecutive order: Question, Sub-Question, Branch-Question. The most important fields of the Results table are listed below. Besides these fields, there are other fields such as Date of Examination, Examination Class Abbreviation, but these are not shown.

The most important fields for the Result table for an examinee:

	Field	Description
1	ExaminationNo2	Examinee's ID
2	BirthDt	Date of Birth
3	SelectionColumn1	Q1 through Q5 are compulsory questions
4	SelectionColumn2	For example: 0010 - the Question 8 is selected, See the item (6)
5	SelectionColumn3	For example: 0010 - the Question 12 is selected, See the item (6)
6	A1	b : Answer b is chosen
7	A2	j : Answer j is chosen
8	A3	b : Answer b is chosen
…	…	…
A+5	ALast	(Chosen Answer for Last Question)

The underlined column's name denotes the primary key.

(4) **CA** table - Correct Answers table. The Correct Answers and the Marks are given for each Branch-Question (or Sub-Question).

	1	2	3	4	5	6
	Question	Sub-Question	Branch-Question	Answers	Marks	CRR
1	1	1	a	a	2.00	
2	1	1	b	d	2.00	
3	1	2		c	4.00	
4	2	1		d,e	2.00	
…	…	…	…	…	…	
A	13	3	…	a	3.33	

Two-answer selection

- 16 -

(5) **Grade** table has N rows; each row has the following format:

	Field	Description
1	ExaminationNo2	Examinee's ID
2	BirthDt	Date of Birth
3	SelectionColumn1	Q1 through Q5 are compulsory questions
4	SelectionColumn2	For example: 0010 - the Question 8 is selected, See the item (6)
5	SelectionColumn3	For example: 0010 - the Question 12 is selected, See the item (6)
6	M1	Marks for A1
7	M2	Marks for A2
8	M3	Marks for A3
…	….	…..
A+5	MLast	(Mark for Last Question)
A+6	Total	Sum of M1, M2, …MLast

(6) Processing the selections: This processing is based on the selections in the selection column on the answer sheet by the examinees for the selective questions Q6 through Q9 and Q10 through Q13. The program only grades the answers whose question numbers are marked selected in the selection column. When question number is not marked selected in that column, the answers are not graded even when the answers are marked on the answer sheet.

Table 1   Selection Column2

SelectionColumn2 *column*	*Selected Question from Q6 – Q9*
1000	Q6
0100	Q7
0010	Q8
0001	Q9

The SelectionColumn2 table determines which question from Q6 – Q9 is selected.

Table 2 SelectionColumn3

SelectionColumn3 *column*	*Selected Question from Q10 – Q13*
1000	Q10
0100	Q11
0010	Q12
0001	Q13

Similarly, the SelectionColumn3 table determines which question from Q10 – Q13 is selected.

(7) The **Processing two-answer selections** subroutine gives the marks according to the following rules:

Possibilities	*How to give the marks*
Marked two answers and both are correct	Full Marks
Marked one or two answers and one is correct	Marks / 2
All other cases	0

[Flowchart 1: Calculate Marks]
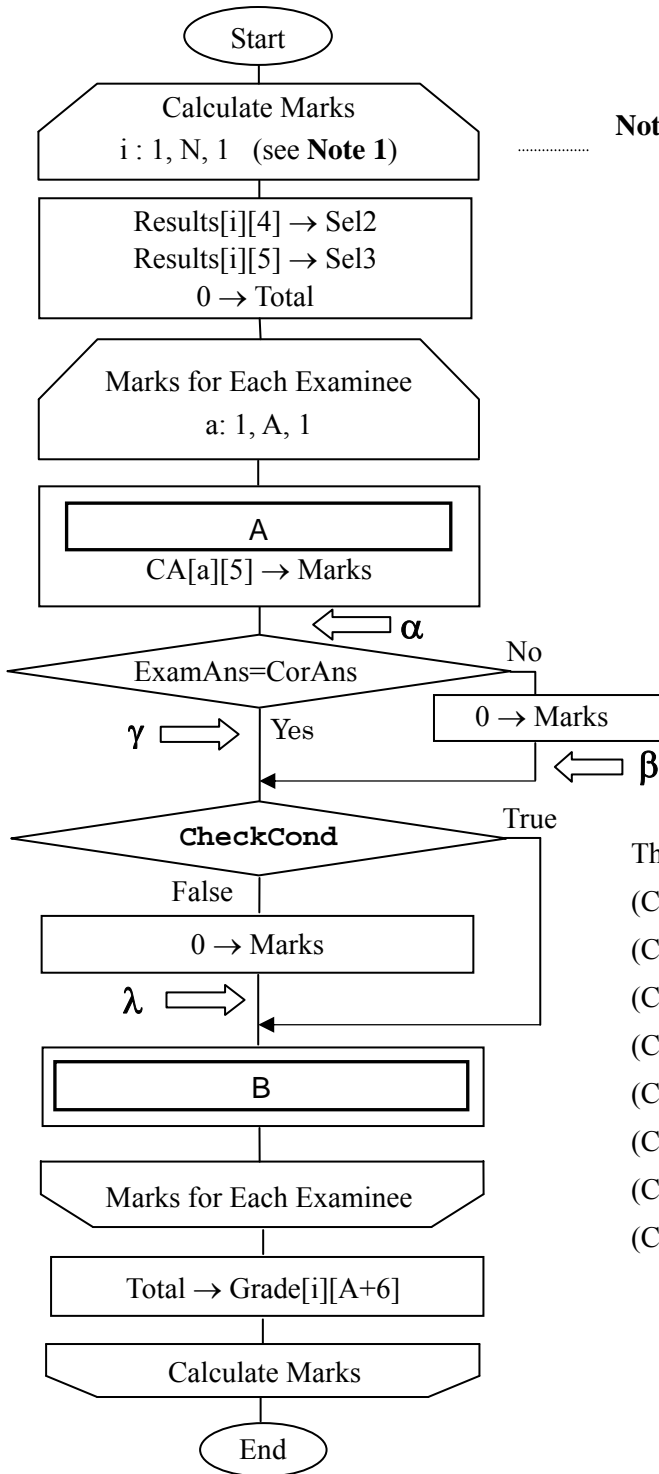
```
                    ┌─────────────┐
                    │    Start    │
                    └─────────────┘
                          │
        ╱─────────────────────────────────╲
        │        Calculate Marks           │
        │  i : 1, N, 1   (see Note 1)      │ ............    Note 1: "1, N, 1" means "from 1 to N by 1"
        ╲─────────────────────────────────╱                        i.e.,  i: 1, 2, …, N
                          │
        ┌─────────────────────────────────┐
        │      Results[i][4] → Sel2        │
        │      Results[i][5] → Sel3        │
        │           0 → Total              │
        └─────────────────────────────────┘
                          │
        ╱─────────────────────────────────╲
        │      Marks for Each Examinee     │
        │            a: 1, A, 1            │
        ╲─────────────────────────────────╱
                          │
        ┌─────────────────────────────────┐
        │  ┌───────────────────────────┐  │
        │  │             A             │  │
        │  └───────────────────────────┘  │
        │        CA[a][5] → Marks         │
        └─────────────────────────────────┘
                          │         ⇐ α
                          │              No
        ╱─────────────────────────────────╲
        ╲       ExamAns=CorAns            ╱──────┐
          ╲─────────────────────────────╱       │
       γ ⇒      │ Yes                     ┌──────────────┐
                │                         │  0 → Marks   │
                │                         └──────────────┘
                │←────────────────────────────┘   ⇐ β
        ╱─────────────────────────────────╲
        ╲          CheckCond              ╱────── True
          ╲─────────────────────────────╱          │
                │ False                             │
        ┌──────────────────────┐                   │
        │      0 → Marks        │                   │
        └──────────────────────┘                   │
         λ ⇒    │←──────────────────────────────────┘
        ┌─────────────────────────────────┐
        │  ┌───────────────────────────┐  │
        │  │             B             │  │
        │  └───────────────────────────┘  │
        └─────────────────────────────────┘
                          │
        ╱─────────────────────────────────╲
        ╲      Marks for Each Examinee     ╱
          ╲─────────────────────────────╱
                          │
        ┌─────────────────────────────────┐
        │      Total → Grade[i][A+6]       │
        └─────────────────────────────────┘
                          │
        ╱─────────────────────────────────╲
        ╲        Calculate Marks           ╱
          ╲─────────────────────────────╱
                          │
                    ┌─────────────┐
                    │     End     │
                    └─────────────┘
```
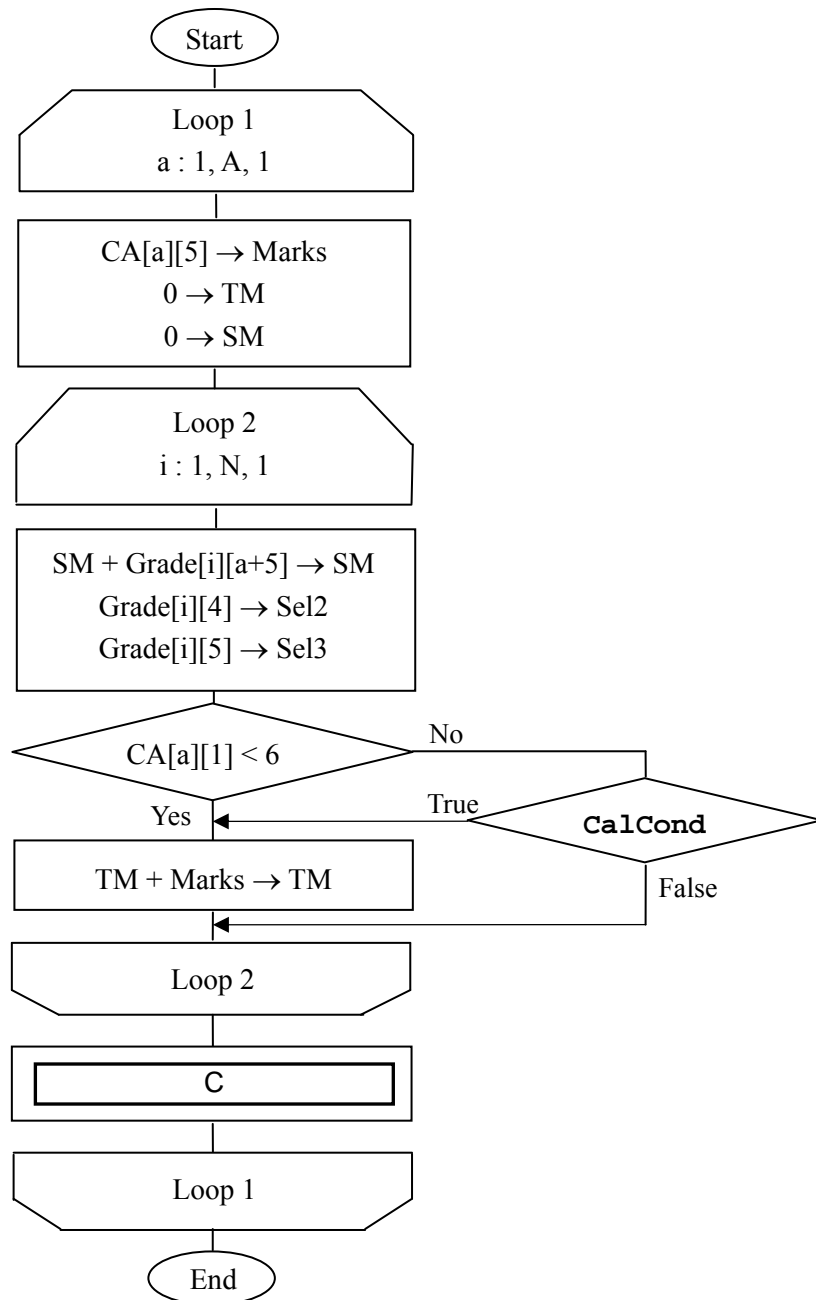
The **CheckCond** is as follows:

(CA[a][1]=6 & Sel2!="1000") OR

(CA[a][1]=7 & Sel2!="0100") OR

(CA[a][1]=8 & Sel2!="0010") OR

(CA[a][1]=9 & Sel2!="0001") OR

(CA[a][1]=10 & Sel3!="1000") OR

(CA[a][1]=11 & Sel3!="0100") OR

(CA[a][1]=12 & Sel3!="0010") OR

(CA[a][1]=13 & Sel3!="0001")

Note 2: This flowchart does not include the

`Processing two-answer selections` subroutine.

[Flowchart 2: Calculate CRR]

For the analysis purposes, a Correct Response Rate (CRR: the percentage of the correct answers per total answers) is calculated for each branch-question (or sub-question).

```
                        ( Start )
                           |
        /-----------------------------------\
        |            Loop 1                  |
        |          a : 1, A, 1               |
        \-----------------------------------/
                           |
        +-----------------------------------+
        |        CA[a][5] → Marks            |
        |           0 → TM                   |
        |           0 → SM                   |
        +-----------------------------------+
                           |
        /-----------------------------------\
        |            Loop 2                  |
        |          i : 1, N, 1               |
        \-----------------------------------/
                           |
        +-----------------------------------+
        |   SM + Grade[i][a+5] → SM          |
        |      Grade[i][4] → Sel2            |
        |      Grade[i][5] → Sel3            |
        +-----------------------------------+
                           |
                      No
        <    CA[a][1] < 6    >------------------------\
                           |                          |
               Yes         True    <   CalCond   >    |
                |<--------------------                |
        +-----------------------------------+   False |
        |        TM + Marks → TM            |<---------/
        +-----------------------------------+
                           |
        \-----------------------------------/
        |            Loop 2                  |
        /-----------------------------------\
                           |
        +-----------------------------------+
        |             C                     |
        +-----------------------------------+
                           |
        \-----------------------------------/
        |            Loop 1                  |
        /-----------------------------------\
                           |
                       ( End )
```

The **CalCond** is as follows:

(CA[a][1] = 6 AND Sel2 = "1000") OR (CA[a][1] = 7 AND Sel2 = "0100") OR
(CA[a][1] = 8 AND Sel2 = "0010") OR (CA[a][1] = 9 AND Sel2 = "0001") OR
(CA[a][1] = 10 AND Sel3 = "1000") OR (CA[a][1] = 11 AND Sel3 = "0100") OR
(CA[a][1] = 12 AND Sel3 = "0010") OR (CA[a][1] = 13 AND Sel3 = "0001")

Subquestion 1

Where the `Processing two-answer selections` subroutine must be inserted in the Flowchart 1?

Answer group
   a) α   b) β   c) γ   d) λ

Subquestion 2

From the answer groups below, select the correct answers to be inserted in the blanks

[   ] in the above flowcharts.

Answer group for A
   a) Results[i][a+4] → ExamAns
      CA [a][4] → CorAns
   b) Results[i][a+4] → ExamAns
      CA [i][4] → CorAns
   c) Results[i][a+5] → ExamAns
      CA [a][4] → CorAns
   d) Results[i][a+5] → ExamAns
      CA [i][4] → CorAns

Answer group for B
   a) Marks → Grade[i][a+4]
      Marks → Total
   b) Marks → Grade[i][a+4]
      Total+Marks → Total
   c) Marks → Grade[i][a+5]
      Marks → Total
   d) Marks → Grade[i][a+5]
      Total+Marks → Total

Answer group for C
   a)   SM / TM → CA[a][6]
   b)   (SM / TM) * 100 → CA[a][6]
   c)   (SM / TM) / 100 → CA[a][6]
   d)   (TM / SM) * 100 → CA[a][6]

Q6 Read the following description of a C program and the program itself, and then answer
the Subquestion.

[Program Description]

A program was created for receiving a null terminated string `s` (an array of characters that is
terminated by '`\0`') and a width `len`, then display the string `s` in the word-wrapped format
of specified width `len`. Assume that no word in the string is longer than the width `len`.

1) Consider a word to be any sequence of non-whitespace characters. The whitespace
characters include newline, horizontal tab, vertical tab, form feed, and space characters.

2) The program uses two functions as follows:

i) The function `cleanText` receives a null terminated string `s` and returns cleaned string
through the same parameter as the original string `s`. The cleaned string contains only one
space character between words and no leading whitespace characters. The function also
reduces all multiple occurences of whitespace characters to a single space character.

ii) The function `wordWrap` receives a cleaned string `s` and a width `len`, then returns the
wrapped string through the same parameter as the original string `s`.

3) The following sample output shows the execution results of the program.

```
Enter a string: This is the course PFC
Enter a width: 10
The wrapped string:
This is
the course
PFC
```

[Program]

```c
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define MAX 100

void cleanText(char s[])
{
  char s1[MAX];
  int inWord = 0, i, j;
  for(i=0, j=0; s[i] != '\0'; i++)
  {
    if(inWord == 0)
      {
          if(          A          )
          {
                inWord = 1;
                if(j != 0) s1[j++] = ' ';
                s1[j++] = s[i];
          }
      }
    else
      {
          if(          B          ) inWord = 0;
          else s1[j++] = s[i];
      }
  }
  s1[j] = '\0';
  strcpy(s, s1);
}

void wordWrap( char s[], int len)
{
   int prev = 0, i, line = 0;

   for(i=0; s[i] != '\0'; i++)
     {
        line ++;
        if (s[i] == ' ')
          {
             if (       C       ) {s[i] = '\n';       D       ;}
             else if (       E       && prev != 0)
                { s[prev] = '\n';       F       ;}

             prev = i;
          }
     }
   if (line > len+1 && prev != 0)
     {
        s[prev] = '\n';
        line = i - prev;
     }
}
```

```
main()
{   char s[MAX];
    int len;

    printf("\nEnter a string:");
    scanf("%[^\n]", s);
    printf("\nEnter a width:");
    scanf("%d", &len);
    cleanText(s);
    wordWrap(s, len);
    printf("\nWraped string:");
    printf("\n%s", s);
    getch();

}
```

Subquestion

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

Answer group for A and B

```
a) s[i]==' '&&s[i]=='\t'&&s[i]=='\v'&&s[i]=='\f'

b) s[i]==' '||s[i]=='\t'||s[i]=='\v'||s[i]=='\f'

c) s[i]!=' '&&s[i]!='\t'&&s[i]!='\v'&&s[i]!='\f'

d) s[i]!=' '||s[i]!='\t'||s[i]!='\v'||s[i]!='\f'
```

Answer group for C through F

```
a) line == len-1              b) line == len+1
c) line > len-1           d) line > len+1
e) line = 0                   f) line = 1
g) line = i - prev            h) line = i - prev + 1
```

- 24 -

Q7 Read the following description of Java programs and the programs themselves, and then answer the Subquestions 1, 2 and 3.

[Program Description]

Program 1 implements the exception handling after an arithmetic operation to find out what kind of exception has occurred when executing the program.

Program 2 tests the inheritance of objects as sample, based on the concept of Object Oriented Programming.

Program 3 shows a sample of string handling technique for Java program.

[Program 1]
```java
package exam;
public class TestException {
    public static void main(String[] args) {
        int k=0;
        try {
            int i = 5/k;
        } catch (ArithmeticException e) {
            System.out.println("1");
        } catch (RuntimeException e) {
            System.out.println("2");
            return;
        } catch (Exception e) {
            System.out.println("3");
        } finally {
            System.out.println("4");
        }
        System.out.println("5");
    }
}
```

[Program 2]
```java
package exam;

class SuperClass {
      public String greeting = "Hello";

      public void displayName(){
            System.out.println(" SuperClass!");
      }
}

class SubClass extends SuperClass {
      public String greeting = "Good Bye";

      public void displayName() {
            System.out.println(" SubClass!");
      }
}

public class TestOverriding {
      public static void main(String[] args) {
            SuperClass obj = new SubClass();

            System.out.print(obj.greeting);
            obj.displayName();
      }
}
```

[Program 3]

```
package exam;

public class StringReplace {

    public static void main(String[] args) {
        String text = "Java is an OOP language. "
                    +"Java is similar to C++.";
        String search = "Java";
        String sub = "C#";
        String result = "";
        int i;

        do{
            System.out.println(text);
            |        a        | ;
            if(i!=-1){
                result = text.substring(0,i);
                result = result+sub;
                |        b        | ;
                text = result;
            }
        }while(i!=-1);
    }
}
```

Output of the program

```
Java is an OOP language. Java is similar to C++.
C# is an OOP language. Java is similar to C++.
C# is an OOP language. C# is similar to C++.
```

Subquestion 1

Read the program 1 and select the correct statement from the answer group below.

Answer group

    a) The program will only print 1 and 4, in that order.

    b) The program will only print 5.

    c) The program will only print 1, 4, and 5, in that order

    d) The program will only print 1, 2, and 4, in that order.

Subquestion 2

When the program 2 is run, what will be the output of the program? Select the correct answer from the answer group below.

Answer group

    a) Good Bye SubClass!

    b) Good Bye SuperClass!

    c) Hello SubClass!

    d) Hello SuperClass!

Subquestion 3

From the answer groups below, select the correct answers to be inserted in the blanks

☐ in the above program 3.

Answer group for a

```
a) i = text.endOf(search)          b) i = text.indexOf(search)
c) i = text.lastIndexOf(search)    d) i = text.startOf(search)
```

Answer group for b

```
a) result = result + text.substring(i+search.length())
b) result = result + text.substring(i+search.length)
c) result = result + text.substring(i+sub.length())
d) result = result + text.substring(i+sub.length)
```

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

[Program Description]

Given the number of goods **N,** positive weight of each good **weight[N]**, positive profit of each good **profit[N]** and a positive number **W** which is the knapsack capacity. This problem is called "the zero-one knapsack" for choosing the most beneficial goods for the knapsack of limited capacity.

Example:
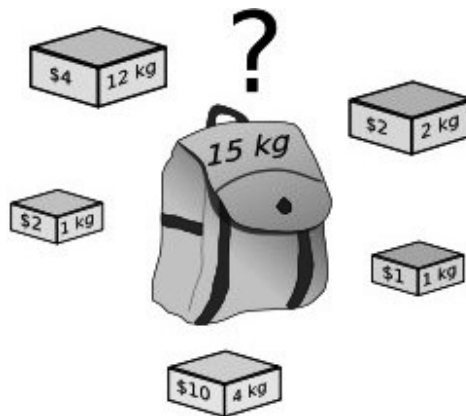
Among the five goods, the four goods $10, 4kg; $2, 1kg; $2, 2kg; $1, 1kg are selected to be the most beneficial for the price of the goods shown in figure above and put them into knapsack. Total profits will be $15 and total weight will be 8kg.

It is considered that data mentioned in [Program Description] have been read from a file.
The following subfunctions, arrays, and variables are used for the program.

goods_into_knapsack()	The function is to put the goods into the knapsack until the knapsack is full.
update_solution()	The function is to store current goods in the knapsack. Updated profit and weight are set to the variables fp and fw.
bound ()	The function is to return the amount of goods that are following the last chosen one.
remove_goods()	The function is to remove the last chosen goods from the knapsack.
X[1:n]	An array to indicate goods in the knapsack. If X[goods]=1, goods with number goods is in the knapsack.
Y[1:n]	An array to temporarily store goods in the knapsack.
cw, curweight	Current total weight of goods in the knapsack.

cp, curprofit	Current total profit of goods in the knapsack.
fw	Final total weight of goods in the knapsack.
fp	Final total profit of goods in the knapsack.

[Program]

```c
#include <stdlib.h>
void goods_into_knapsack();
void update_solution();
void remove_goods(int *flag);
float bound(int last_removed);

int goods, N, *Y, *X;
float *weight, *profit;
float cw = 0, cp = 0, fw, fp, W;

void main()
  {

// Note that data have already been read in from a file.

// Prepare memory for arrays.
  weight = (float*) malloc (N*sizeof(float));
  profit = (float*) malloc (N*sizeof(float));
  Y = (int*) malloc (N*sizeof(int));
  X = (int*) malloc (N*sizeof(int));

  int flag = 0 ;
  goods = 0 ;
  fp = -1 ;
  while (!flag)
    {
      // Place goods into knapsack
      goods_into_knapsack();

      // update solution
      update_solution();

      while ( bound(goods) <= fp )
      {
         remove_goods(&flag);
         if(flag)
            break;
```

```
          }
         if(!flag)
        goods++;

      } // end of while
    } //end of main


// Place goods into knapsack
void goods_into_knapsack()
   {  // Iteration to put goods into the knapsack.
      While ( goods  [ A ]  N  [ B ]  (cw + weight[goods]  [ C ]  W))
      {
       cw += weight[goods];
       cp += profit[goods];
       Y[goods] = 1;
       goods++;
      }
   }


   // update solution
  void update_solution()
   {
      int i;
      if (    [   D   ]    )
      {
       fp = cp;
       fw = cw;
       goods = N-1;
       for (i=0; i < N; i++)
          X[i] = Y[i];
      }
      else
        Y[goods] = 0;
   }


void remove_goods(int *flag)
   {  // Iteration to find the good to be removed.

      while (   [    E    ]   )
         goods--;

      if (goods == -1)
      {
        *flag = 1;
        return ;
      }
```

```
     Y[goods] = 0;
    cw -= weight[goods];
    cp -= profit[goods];
   }


float bound(int last_removed)
   {
      int i ;
      float b = cp ;
      float c = cw ;

      for (i = last_removed + 1; i < N; i++)
      {    c += weight[i] ;
          if (c <= W)
               b += profit[i] ;
         else
          {
               b+= (1-(c-W)/weight[i])*profit[i] ;
               break ;
          }
       }
        return b ;
    }
```

Subquestion

From the answer groups below, select the correct answers to be inserted in the blanks

| | in the above program.

Answer group for A, B and C

a) `>`

b) `>=`

c) `==`

d) `<=`

e) `<`

f) `&&`

g) `||`

Answer group for D

a) `goods > N`

b) `goods > N-1`

c) `goods < N`

d) `goods < N-1`

e) `goods == N-1`

Answer group for E

a) `goods != -1 || Y[goods]`

b) `goods || Y[goods]`

c) `goods != -1 || Y[goods]!=1`

d) `goods || Y[goods]!=1`

e) `goods != -1 && Y[goods]!=1`

f) `goods && Y[goods]!=1`

g) `goods != -1 && Y[goods]`

h) `goods && Y[goods]`

Q9 Read the following description of Java programs and the programs themselves, and then answer the Subquestions 1 through 4.

This is a complete Vehicle Registration Program that will register new vehicles. When registering a new Truck, the data that will be kept is a tag, name of the person registered to and the company that owns the truck. Information on the empty weight of the truck, the last odometer and the type of the truck (PICKUP or BIGRIG) is also obtained when created the truck object.

Classes and their explanation are shown below:

Program 1: RegisteredVehicle	This is the highest class in the hierarchy. Defines 3 data – tag, registeredTo and owner. Has a constructor and 2 methods – reRegister and printRegistration. Also has an abstract method named getAnnualFee.
Program 2: RegistrationException	For displaying error messages if any errors occur when reregistration of a truck is done.
Program 3: RegisteredMotorVehicle	Second class in the hierarchy. Adds on two more data – emptyWeight and lastOdometer. It has a constructor of its own but two methods are overridden from the parent class – reRegister and printRegistration.
Program 4: Truck	Third class in the hierarchy. Defines truckType which is either a PICKUP or a BIGRIG. It has a constructor of its own but a method is overridden from the parent class – printRegistration.
Program 5: RegisteredVehicleTest	This is the executable class, where two trucks are created into a RegisteredVehicle array.

Methods used in the program and their explanation are shown below:

reRegister	Updates information on who the vehicle is registered to and the owner of the vehicle
printRegistration	Displays information on vehicles
getAnnualFee	Calculates the annual fee by the empty weight of the trucks categorized by its type.

[Program 1]
```
1.  abstract class RegisteredVehicle {
2.     String tag ;
3.     String registeredTo ;
4.     String owner ;
5.     [    A    ](String t, String r, String o) {
6.       tag = t ;
7.       registeredTo = r ;
8.       owner = o ;
9.     }
10.    void reRegister(String r, String o) {
11.      registeredTo = r ;
12.      owner = o ;
13.    }
14.    void printRegistration() {
15.      System.out.println("Tag:  " + tag) ;
16.      System.out.println("Registered to:  " + registeredTo);
17.      System.out.println("Owner:  " + owner) ;
18.    }
19.    abstract int getAnnualFee();
20. }
```

[Program 2]
```
1.  class RegistrationException [    B    ] {
2.     RegistrationException(String s) { super(s) ; }
3.  }
```

[Program 3]
```
1.  [   C   ] class RegisteredMotorVehicle extends [    A    ] {
2.     int emptyWeight ;
3.     int lastOdometer ;
4.     RegisteredMotorVehicle(String t,String r,String o,int w, int l) {
5.       super(t, r, o) ;
6.       emptyWeight = w ;
7.       lastOdometer = l ;
8.     }
9.     void reRegister(String r, String o, int l)
10.      throws RegistrationException {
```

- 35 -

```
11.        if (l < lastOdometer)
12.          throw new RegistrationException("Invalid odometer reading.");
13.          [    D    ] ;
14.          lastOdometer = l ;
15.        }
16.    void printRegistration() {
17.      super.printRegistration() ;
18.      System.out.println("Empty weight:  " + emptyWeight);
19.      System.out.println("Last odometer reading:  " + lastOdometer) ;
20.    }
21. }
```

[Program 4]

```
1.  class Truck extends [      E      ] {
2.    static final int PICKUP = 0, BIGRIG = 1 ;
3.    int truckType ;
4.    Truck(String t,String r,String o,int w,int l,int type) {
5.      super(t, r, o, w, l) ;
6.      truckType = type ;
7.    }
8.    void printRegistration() {
9.      super.printRegistration() ;
10.      switch (truckType) {
11.        case PICKUP : {
12.          System.out.println("Pickup") ;
13.          break ;
14.        }
15.        case BIGRIG : {
16.          System.out.println("Big rig") ;
17.          break ;
18.        }
19.        default :
20.        System.out.println("Truck") ;
21.      }
22.    }
23.    int getAnnualFee() {
24.      int temp ;
25.      switch (truckType) {
```

```
26.       case PICKUP : {
27.          temp = (int) (emptyWeight * .05) ;
28.          break ;
29.       }
30.       case BIGRIG : {
31.          temp = (int) (emptyWeight * .01) ;
32.          break ;
33.       }
34.       default :
35.          temp = (int) (emptyWeight * .10) ;
36.     }
37.     return temp ;
38.   }
39. }
```

[Program 5]

```
1. public class RegisteredVehicleTest {
2.    public static void main(String[] args) {
3.       RegisteredVehicle[] vs = {
4.          new Truck("1234XYZ", "J. Smith", "Big Bank",
5.                    2000, 20000, Truck.PICKUP),
6.          new Truck("2345YXZ", "B. Jones", "Truckers CU",
7.                    10000, 40000, Truck.BIGRIG)
8.       } ;
9.       for (int i = 0 ; i < vs.length ; i++) {
10.        vs[i].printRegistration() ;
11.        System.out.println("Annual fee: $" + vs[i].getAnnualFee());
12.        System.out.println("");
13.      }
14.      try {
15.        RegisteredMotorVehicle r = (RegisteredMotorVehicle)vs[0];
16.        r.reRegister("J. Smith", "J. Smith", 40000) ;
17.      }
18.      catch (RegistrationException e) {
19.        System.out.println(e.getMessage()) ;
20.      }
21.      System.out.println("") ;
22.      for (int i = 0 ; i < vs.length ; i++) {
```

```
23.          vs[i].printRegistration() ;
24.          System.out.println("Annual fee:  $" + vs[i].getAnnualFee());
25.          System.out.println("") ;
26.      }
27.   }
28. }
```

Subquestion 1

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

Answer group for A and E
    a) RegisteredMotorVehicle
    b) RegisteredVehicle
    c) RegisteredVehicleTest
    d) RegistrationException
    e) Truck

Answer group for B
    a) extends Exception
    b) extends Object
    c) extends RegisteredVehicle
    d) extends String

Answer group for C
    a) abstract
    b) private
    c) protected
    d) public
    e) void

Answer group for D
    a) reRegister(w, l)
    b) super.reRegister(r, o)
    c) super.reRegister(w, l)

Subquestion 2

In the class `RegisteredMotorVehicle`, why isn't the method `getAnnualFee()` declared?

Answer group

    a) Declaring `getAnnualFee()` in `RegisteredVehicle` will display a `RunTimeException()` error

    b) Declaring `getAnnualFee()` in `RegisteredVehicle` will override it and therefore cannot be declared again

    c) `getAnnualFee()` can only be inherited once as given on Line 23 in Program 4

    d) You do not need to declare `getAnnualFee()` in the `RegisteredMotorVehicle` class

Subquestion 3

Line 14 in Program 1 and Line 16 in Program 3 shows:

Answer group

    a) method abstraction

    b) method inheritance

    c) method overloading

    d) method overriding

    e) method polymorphism

Subquestion 4

What does Line 15 in Program 5 do?

Answer group

    a) The object `vs[0]` is casted as a `RegisteredMotorVehicle` object as the method `reRegister` is not available in the `RegisteredVehicleTest` class.

    b) The object `vs[0]` is casted as a `RegisteredMotorVehicle` object as the method `reRegister` is not available in the `RegisteredVehicle` class.

    c) The object `vs[0]` is compared to the object `r` from the class `RegisteredMotorVehicle`.

    d) The object `vs[0]` is created as a new object of type `RegisteredMotorVehicle`.