Spring 2000

Fundamental Information Technology Engineer Examination (Afternoon)

(former CLASSⅡ)

Questions must be answered in accordance with the following:

Question Nos.	**Q1 – Q5**	**Q6 – Q8**	**Q9 – Q11**
Question Selection	Compulsory	Select 1 of 3	Select 1 of 3
Examination Time	13:00 ~ 15:30		150 minutes

Instructions:

1. Use an HB pencil.
   If you need to change an answer, erase your previous answer completely and neatly. Wipe away any eraser debris.

2. Mark your answers in accordance with the instructions below. Your answers will not be graded if you fail to comply with the instructions. 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 exam ticket, and mark the appropriate space below each digit.

(3) Question Selection

Mark the questions that you select to answer as follows: [ ].

(4) Answers

Mark your answer as shown in the following sample question:

How to Mark Your Answers

Right	Wrong			
■	⊘	Faint	✓	□

[Sample Question] Select the correct answer to be inserted into the corresponding box (☐) from the choices provided below.

The Spring Information Technology Engineer Examination is held in ☐a☐.

Answer group for a:

a) April b) May c) June

Since the correct answer is "A" (April), mark your answer sheet as follows:

[Sample Reply]

a ■ [b.] [c.]

3. The specifications of an Assembler language are provided as a reference at the end of this booklet. The information provided on page 70 and beyond, however, is only intended for use with Q17.

Do not open the exam booklet until instructed to do so.
Inquiries about the exam questions will not be answered.

Q1 and Q5 below are compulsory. Answer all sub-questions.

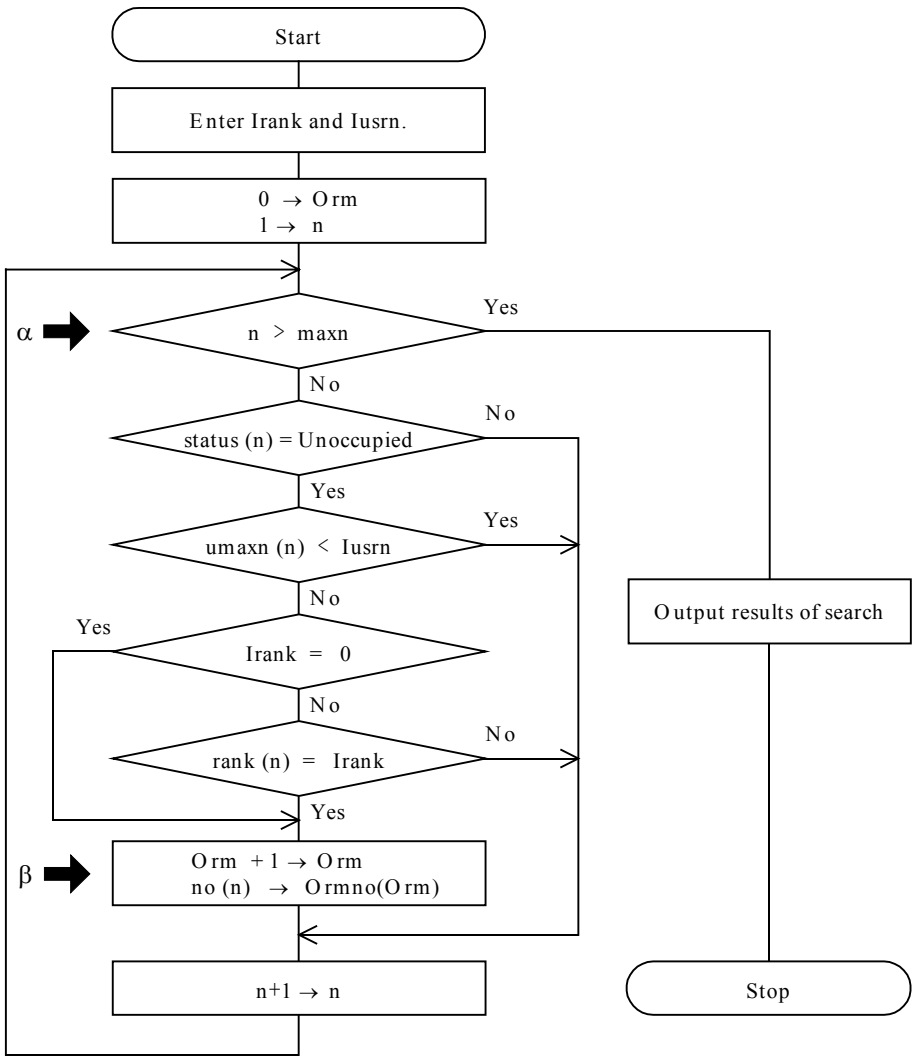
Q1. Examine the flow chart that follows, and read the explanation below before answering the sub-questions.

[Explanation of Flow Chart]
This flow chart searches for unoccupied rooms in a hotel. It accepts the room rank and number of occupants as input and then outputs the number of unoccupied rooms and the corresponding room numbers. Use zero for the room rank if it is irrelevant. Table 1 gives the main variables used in the flow chart. Subscript n, used with the array, takes values in the range 1 through maxn. Subscript m takes values 1 through Orm.

Table 1 Main Variables

Variable names	Description	Range of values
maxn	Number of rooms	–
no(n)	Room number	–
umaxn(n)	Maximum number of occupants of room	–
status(n)	Room status	Unoccupied Occupied
rank(n)	Room rank	1 through 3
Irank	Room rank to be found	0 through 3
Iusrn	Number of occupants of room to be found	–
Orm	Number of unoccupied rooms	–
Ormno(m)	Available room numbers	–

[Flow Chart]


Start
Enter Irank and Iusrn.
0 → Orm
1 → n
α
n > maxn
Yes
No
status (n) = Unoccupied
No
Yes
umaxn (n) < Iusrn
Yes
No
Irank = 0
Yes
No
rank (n) = Irank
No
Yes
β
Orm + 1 → Orm
no (n) → Ormno(Orm)
n+1 → n
Output results of search
Stop

Sub-Question 1

Select the correct answers to be inserted into the following boxes ( ) from the choices provided.

When the room status is as given in Table 2, and a search is executed using the following input data, the value of Orm is [a] and the value of Ormno(2) is [b].

Irank : 0 Iusrn : 4

Table 2 Room Configuration and Status

n	no	umaxn	status	rank
1	101	4	Unoccupied	3
2	102	4	Occupied	3
3	201	3	Unoccupied	2
4	202	4	Occupied	2
5	301	6	Unoccupied	1

Answer group for a and b:

a) 1	b) 2	c) 3	d) 4	e) 5
f) 101	g) 102	h) 201	i) 202	j) 301

Sub-Question 2

Add to the flow chart the process which calculates the nightly room rate for the number of users and stores this value in the array Ormp. Select from the answer group below the correct answer to be added after the box labeled β in the flow chart. The rate per occupant for each room rank (Table 3) is stored in the array rct. The subscript used for the array rct represents the room rank starting from 1.

Table 3 Rates (rct)

Room rank	Rate (yen/person)
1	20,000
2	10,000
3	8,000

Answer group:

a) Iusrn → Ormp(Orm)
b) rct(Irank) → Ormp(n)
c) rct(Irank) + Iusrn → Ormp(n)
d) rct(Irank) × Iusrn → Ormp(n)
e) rct(Irank) × Iusrn → Ormp(Orm)
f) rct(rank(n)) × Iusrn → Ormp(n)
g) rct(rank(n)) × Iusrn → Ormp(Orm)

Sub-Question 3

Add variable rcmaxn to give the maximum number of output items so that the number of output items resulting from a search can be limited. How should the conditional statement in the lozenge-shaped box labeled α be changed? Select the correct answer from the choices provided below.

Answer group:

a) n < maxn and Orm < rcmaxn
b) n < maxn and Orm ≥ rcmaxn
c) n > maxn and Orm > rcmaxn
d) n < maxn or Orm > rcmaxn
e) n > maxn or Orm < rcmaxn
f) n > maxn or Orm ≥ rcmaxn

Q2. Read the following description related to pixels displayed on a screen and then answer the sub-questions.

The screen size is 128 pixels in both the horizontal and vertical directions. The x coordinate represents the horizontal direction, while the y coordinate represents the vertical direction. An array named MAP (subscript beginning at) is used to store 1 bit to indicate the state of each pixel on the screen. The size of each element in the array MAP is 16 bits. When the value of each bit is "1", the corresponding pixel is turned on. When "0", the pixel is turned off. The figure below shows the relationship between the coordinates of pixels on the screen and the position of bits in elements in the array MAP.

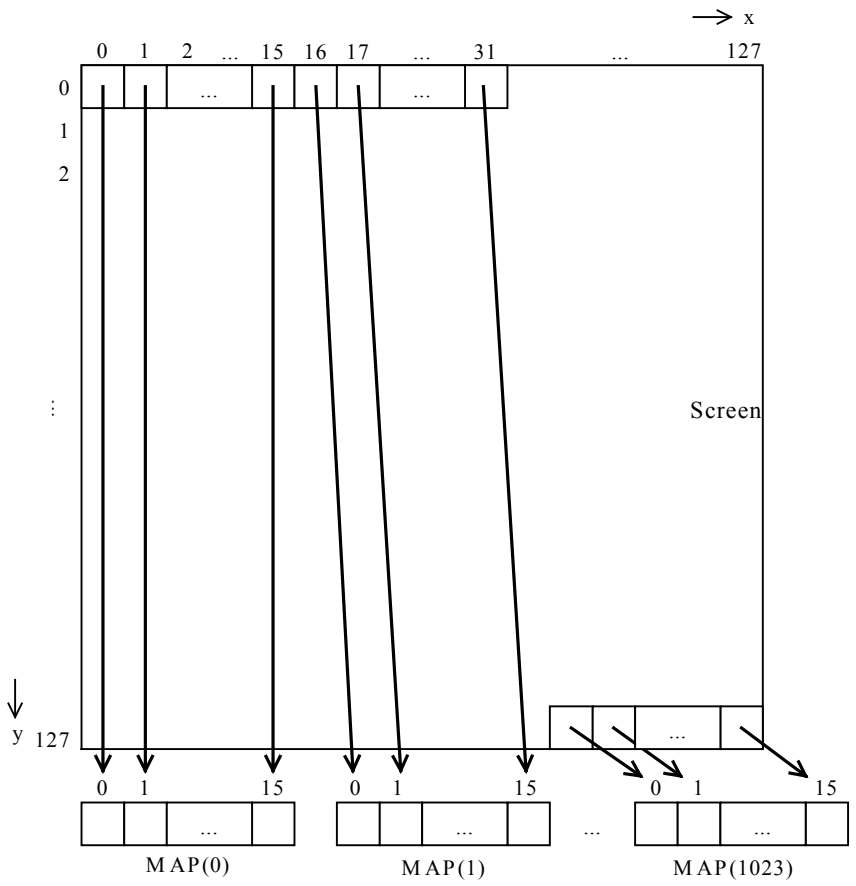


Fig. Correspondence between Pixel Coordinates and the Array MAP

Sub-Questions 1, 2 and 3

Create an algorithm for turning the corresponding pixel on or off when coordinates (x, y) are given. Select the correct answers from the answer groups below to be inserted into the corresponding boxes ( ) in the following description of this algorithm. Assume that X, Y, N, V, Q, S and W are all 16-bit, unsigned integers.

(1) Find number V of the array element that includes the bit that corresponds to the pixel at coordinates (x, y).

① Assign x to X and y to Y.

② Assume the number of array elements corresponding to a single horizontal line on the screen is N. To multiply Y by N, arithmetically shift Y by [a] and assign the result to Y.

③ Arithmetically shift X by [b] and assign the result to S.

④ Calculate Y + S and assign the result to V.

(2) Find data Q where only the appropriate bit of the array element MAP (V) corresponding to the element at (x, y) is 1. To do this, access the array BIT, described in the table below.

Table Contents of the Array BIT

Array element	Element value	Array element	Element value
BIT(0)	32768	BIT(8)	128
BIT(1)	16384	BIT(9)	64
BIT(2)	8192	BIT(10)	32
BIT(3)	4096	BIT(11)	16
BIT(4)	2048	BIT(12)	8
BIT(5)	1024	BIT(13)	4
BIT(6)	512	BIT(14)	2
BIT(7)	256	BIT(15)	1

The procedure for finding Q is as follows.

① Find the [c] of X and 15 and assign the result to X.

② Assign the value of BIT(X) to Q.

(3) Change the contents of the MAP array to turn the pixel at coordinate (x, y) either on or off.

① Assign the value of MAP(V) to W.

② To turn a pixel on, find the [d] of W and Q and assign the result to MAP(V).

③ To turn a pixel off, reverse all the bits of Q (exchange 0's for 1's), find the logical product of the bits of W and Q, and assign the value to MAP(V).

Answer group for a and b:

a) Shift 1 bit left
b) Shift 2 bits left
c) Shift 3 bits left
d) Shift 4 bits left
e) Shift 1 bit right
f) Shift 2 bits right
g) Shift 3 bits right
h) Shift 4 bits right

Answer group for c and d:

a) 2's complement
b) Bitwise exclusive logical sum
c) Bitwise logical product
d) Bitwise logical sum
e) Reverse bits
f) Additive sum
g) Subtractive sum
h) Product

Q3. Read the following explanation of a scheduler and then answer the sub-questions.

[Description of Scheduler]

(1) The scheduler implements multiple programming using a single processor.
(2) A time slice is 50 milliseconds.
(3) Processes have no priority level, but are executed using the round-robin method.
(4) If I/O occurs during execution of a process, the next process is executed. Also, if the turn comes to execute a process waiting for I/O, that process is skipped, and the next process is executed.

Sub-Questions 1 and 2

Select the correct answers from the answer groups below to be inserted into the corresponding boxes (☐) in the following description.

There are three processes: P1, P2 and P3. The figure shows the sequence in which the processor and I/O devices are used, and the amount of time during which they are used, when each process is executed independently.

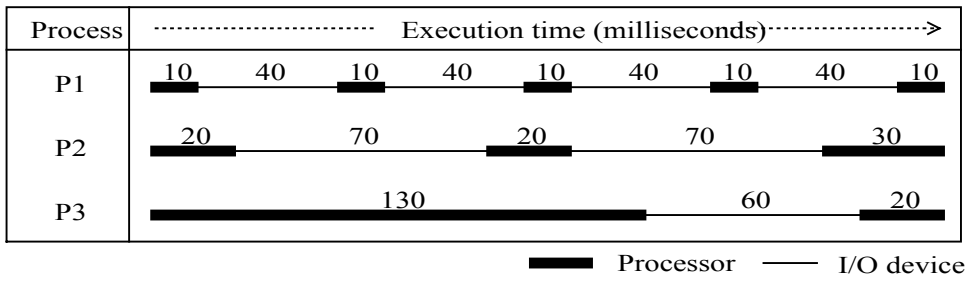


Fig. Sequence and Time Length of Use of the Processor and I/O Devices

Now, assume that the three processes P1, P2 and P3 are executed in round-robin in the sequence P1 → P2 → P3 → P1 → ... Ignore overhead caused by switching between processes.

(1) If the three processes are using different I/O devices α, β and γ respectively, the process that terminates first is [a] and the time in milliseconds from the start of P1 to the end of all the processes have terminated is [b].

(2) If process P1 uses I/O device α, and processes P2 and P3 both use I/O device β (the other waits until the first to begin using the I/O device is finished), then the time until termination is longer than in (1) for process(es) [c] and the amount of time that termination is delayed is [d] milliseconds.

Answer group for a and c:

a) P1 b) P1 and P2 c) P1 and P3
d) P2 e) P2 and P3 f) P3

Answer group for b:

a) 250 b) 260 c) 270 d) 280
e) 290 f) 300 g) 310 h) 320

Answer group for d:

a) 10 b) 20 c) 30 d) 40
e) 50 f) 60 g) 70 h) 80

Q4. Read the following explanation for communication and data compression and answer Sub-Questions 1 and 2.

When transferring data over a communications line, the amount of data to be transferred is reduced using data compression in order to improve transmission efficiency. Huffman encoding is a typical example of the type of data compression used for this.

In the Huffman encoding method, the fact that characters appearing in text data do not appear at the same frequency is utilized and characters with a high frequency of appearance (frequency of occurrence) are replaced with a short bit pattern, while those with a low frequency of appearance are matched with longer bit patterns.

Consider processing that transfers text data represented by single byte (8-bit) characters by replacing them according to Huffman encoding having the following specification.

f) Assume there are 30 types of characters that may be included in text data to be transferred. (The triangle represents a space.)
f) Each character is encoded based on the frequency of occurrence given in the table for that English character.

Table Frequency of Occurrence and Codes for Characters

Character	Frequency of occurrence (%)	Total frequency of occurrence (%)	Huffman code	Character	Frequency of occurrence (%)	Total frequency of occurrence (%)	Huffman code	Character	Frequency of occurrence (%)	Total frequency of occurrence (%)	Huffman code
△	19.0	19.0	000	D	3.1	76.5	110010	B	1.2	96.3	111100
E	9.6	28.6	001	C	3.0	79.5	110011	.	1.0	97.3	111101
T	7.3	35.9	010	L	2.9	82.4	110100	V	0.9	98.2	11111000
A	6.5	42.4	011	M	2.6	85.0	110101	,	0.9	99.1	11111001
O	5.8	48.2	1000	P	2.3	87.3	110110	K	0.3	99.4	11111010
I	5.6	53.8	1001	U	2.1	89.4	110111	X	0.2	99.6	11111011
N	5.5	59.3	1010	F	1.8	91.2	111000	J	0.1	99.7	11111100
R	5.4	64.7	1011	G	1.3	92.5	111001	Q	0.1	99.8	11111101
S	5.1	69.8	110000	W	1.3	93.8	111010	Z	0.1	99.9	11111110
H	3.6	73.4	110001	Y	1.3	95.1	111011	?	0.1	100.0	11111111

Sub-Question 1

Select the bit pattern that will NOT result from using Huffman encoding as described in the table a bove.

Answer group:

a) 010100011010101101010000 b) 011110110110110110100001
c) 110110100001001101010000 d) 111100011101000111010011

Sub-Question 2

Select the correct answers form the answer groups below to be inserted into the corresponding boxes (☐) in the following description.

f) When encoding the English message "HOW△ARE△YOU?", the compression rate for the text data is approximately [a] %.

f) Given the frequency of occurrence for characters in the table, when transferring text data consisting of 5,000 characters, the expected value for the length of data to be transferred is [b] bytes.

Answer group for a:

a) 44 b) 45 c) 55 d) 56 e) 58
f) 60 g) 71 h) 88 I) 89 j) 96

Answer group for b:

a) 2,000 b) 2,168 c) 2,710 d) 2,933 e) 3,667
f) 5,866 g) 21,680 h) 27,100 I) 29,333 j) 36,667

Q5. Read the following explanation of a detail design and then answer Sub-Questions 1 through 4.

Company M has decided to manage the flow of people in and out of their office as shown in Fig. 1, and to implement an access management system that uses IC cards for authorization to enter.

A data storage sub-system and movement status display sub-system are currently being designed.

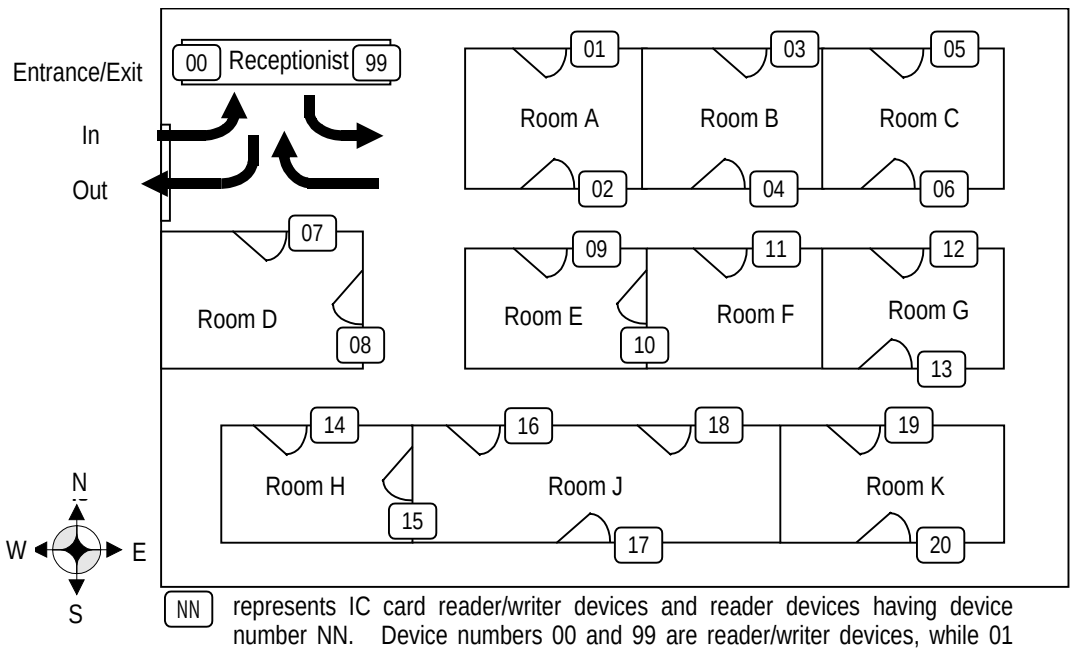


Fig. 1 Office Controlled by Access Management

[Description of Data Storage Sub-System]
Data regarding the entrance and exit of visitors is stored in a relational database as follows.

Fig. 2 shows the configuration of this sub-system.

① The receptionist inserts an IC card issued to the visitor into reader/writer device 00.
② The sub-system reads the entrance verification number specific to that IC card, obtains the reception time from the sub-system clock, and writes the data on the IC card. Since the same IC card is used repeatedly by this sub-system, the generated data is distinguished by the entrance verification number and reception time.
③ The receptionist asks the visitor for necessary information (name, company name, address, telephone number, reason for visit, room to be visited, and person to be visited) and enters it into the sub-system.
④ The sub-system takes the data entered in ③ and the entrance verification number and reception time and uses the information to create a "Table of Visitors", "Table of People Visited", and a "Table of Recorded Data".
⑤ Carrying the IC card issued, the visitor goes to the room of the person to be visited.
⑥ A reader device is installed at the entrance to each room in the office. The sub-system reads the information written on the IC card when the visitor enters and leaves the room, gets the time from the sub-system clock, and writes this data in the "Table of Recorded Data".
⑦ When leaving the office, the visitor returns the IC card to the receptionist. The receptionist then inserts the returned IC card into reader/writer device 99.
⑧ The sub-system gets data from the IC card and the sub-system clock, and writes the data in the "Table of Recorded Data". The IC card is then returned to initial status by writing blank spaces into the reception time. The entrance verification number specific to the IC card remains unchanged.

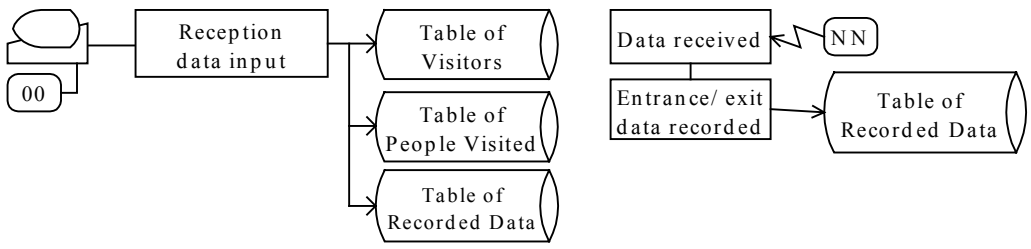


Fig. 2 Configuration of Data Recording Sub-System

[Explanation of Tables and Data Used by the Data Recording Sub-System]

The data format of the IC card and the format of each table are as follows:

Data format of IC card

Entrance verification number	Reception time

Format of Table of Visitors

Entrance verification number	Reception time	Name	Company name	Address	Telephone number

Format of Table of People Visited

Entrance verification number	Reception time	Room being visited	Person being visited	Reason for visit

Format of Table of Recorded Data

Entrance/exit time	Device number	Entrance verification number	Reception time

[Explanation of Movement Status Display Sub-System]

This sub-system creates a record of movement within the office from the time a visitor enters until he/she leaves, and displays the information. Fig. 3 shows the configuration of the movement status display sub-system, while Fig. 4 shows the movement status display screen.

The sub-system:

① Takes the visitor's name entered from a terminal.
② Dynamically generates and executes SQL code 1.
③ Displays a list of data accepted from visitors by name entered for them on a terminal.
④ Reads the reception time and entrance verification number for the entered data as selected from a list.
⑤ Dynamically generates and executes SQL code 2.
⑥ Dynamically generates and executes SQL code 3.
⑦ Edits the data obtained in ⑤ and ⑥ above and displays it on the movement status screen.

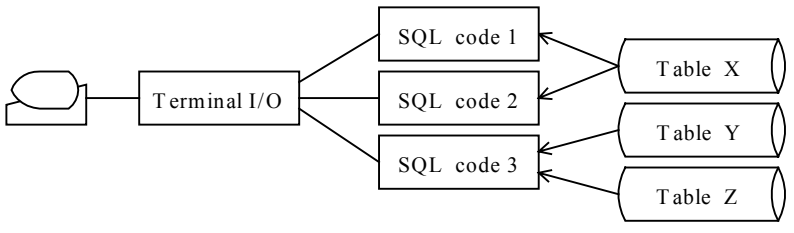


Fig. 3 Configuration of the Movement Status Display Sub-system

Movement Status of Visitors

Entrance verification number: PH007 Name: Yamada, Tarou Company name: JITEC
Address: 1234 Toranomon, Minato-ku, Tokyo Telephone number: 03-8888-9999

Date 04 01 2000

Time	Device number	Device location
09:00	00	Reception (In)
09:05	14	Room H (North)
10:30	15	Room H (East)
10:50	17	Room J (South)
11:00	06	Room C (South)
12:00	05	Room C (North)
12:10	99	Reception (Out)

Fig. 4 Movement Status Display Screen

[Explanation of Table Used by Movement Status Display Sub-System]
In order for the movement status display sub-system to display Fig. 4, it is necessary to use a new table in addition to two of the tables used by the data recording sub-system.

The format for Table Z to be newly added to the movement status display sub-system is as follows.

Format of Table Z

a	b

Note) a is the primary key.

Sub-Question 1

Select the correct answers from the answer groups below to be inserted into the corresponding boxes ( ) above for the definition of the format of Table Z.

Answer group:

a) Reception time
b) Company name
c) Name
d) Device location
e) Device number
f) Entrance/exit time
g) Entrance verification number
h) Person visited
i) Room being visited

Sub-Question 2

Select the correct answer for both Table X and Table Y to appear in Fig. 3 above.

Answer group:

	Table X	Table Y
A	Table of Data	Table of People Visited
B	Table of Data	Table of Visitors
C	Table of People Visited	Table of Data
D	Table of Visitors	Table of Data

Sub-Question 3

Select the correct combination of lines from the choices below for the "Table of Data" to be inserted for the SQL code for Table 3. '`XXX...X`' represents the entrance verification number obtained in ④ of the explanation of the movement status display sub-system, while '`YYY...Y`' represents the reception time.

Answer group:

a)
```
SELECT   Entrance/exit time, Device-number, Device-location FROM
         Table-of-Data
 WHERE   Table-of-Data.Entrance-verification-number =
         'XXX...X' AND Table-of-Data.Reception-time =
         'YYY...Y'
 ORDER   BY Entrance/exittime ASC
```

b)
```
SELECT   Entrance/exit time, Device-number, Device-location FROM
         Table-of-Data
 ORDER   BY Entrance-verification-numberASC, Entrance/exit
         time ASC
```

c)
```
SELECT   Entrance/exit time, Device-number FROM Table-of-
         Data
 WHERE   Table-of-Data.Entrance-verification-number=
         'XXX...X' AND Table-of-Data.Reception-time =
         'YYY...Y'
 ORDER   BY Entrance/exittime ASC
```

d)
```
SELECT   Entrance-verification-number, Name, Company-name,
         Address, Telephone-number FROM Table-of-Data
 WHERE   Table-of-Data.Entrance-verification-number =
         'XXX...X' AND Table-of-Data.Reception-time =
         'YYY...Y'
```

Sub-Question 4

There is a chance that problems will occur if the program is designed to obtain reception time values and entrance/exit time values for each table, and the IC card from the clocks in the reader/writer devices and reader devices. Select the correct answer that describes this problem from the choices provided below.

Answer group:

a) The entrance/exit time value in the "Table of Data" differs from the value obtained from the clock in the device having the corresponding device number.
b) The movement status display is not displayed in the sequence that the visitor has actually moved.
c) The value for the reception time for a single visit differs from that on the IC card and in the "Table of Visitors" and "Table of People Visited", even though it should be the same.
d) The value for the reception time for a single visit differs from that in the "Table of Data", even though it should be the same.

Select one question from Question 6 through Question 8, and mark [ ] which question you have selected on your answer sheet.
If you select more than one question, only the first selected question will be graded.

Q6. Read the explanation of the following C program and then answer the sub-question.

[Description of Program]
A bitmap screen has dimensions of 512 pixels in both the vertical and horizontal directions. The X coordinate represents the horizontal direction, while the Y coordinate represents the vertical direction. Coordinates are assigned as shown in the figure below. The function `DisplayClock` has been created to draw a clock face with the specified time on this screen.

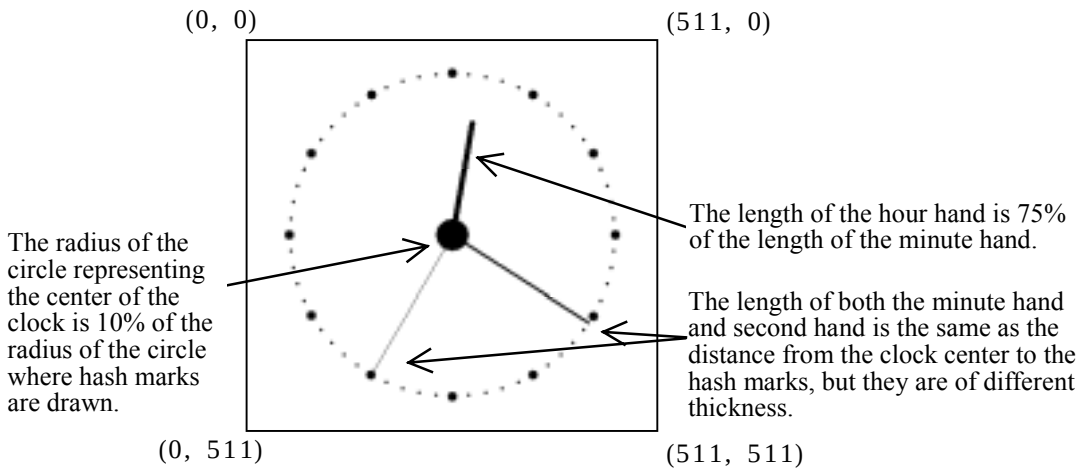


Fig. Coordinate Assignment for the Bitmap Screen

There are 60 hash marks around the circle. Every fifth hash mark is drawn with a large circular marker having a 6-pixel radius, while the other hash marks are drawn with a small circular marker having a smaller 2-pixel radius. Specifications for the clock's hands are as given in the table below.

Table Clock Hand Specifications

	Length	Thickness	Accuracy
Second hand	Distance from center to hash marks	1 pixel	6 degrees (1-second ticks)
Minute hand	Distance from center to hash marks	3 pixels	1 degree (1/10th-second ticks)
Hour hand	75% of distance from center to hash marks	7 pixels	1 degree (2-minute ticks)

The function DisplayClock takes a pointer to the structure ACLOCK (shown below) as an argument.

```
typedef   struct  {
  int Ready;   /* Does not re-calculate coordinate values for hash marks when true.  */
 int Hour, Minute, Second;    /* Time to be drawn.                                    */
 int Cx, Cy;  /* Coordinate value of rotational axis for clock hand.                  */
 int Radius;  /* Radius from center to hash marks.                                    */
 int PlateX[360], PlateY[360];  /* Hash mark coordinates in 1-degree units.          */
} ACLOCK;
```

The ACLOCK structure variable stores the coordinate value for the clock center, the radius of the hash mark circle, and its coordinate values. The value of Ready, a member variable of the ACLOCK structure, is FALSE when the clock is first drawn, when the clock is moved, or when the clock's size is changed. If the program is executed when Ready is FALSE, coordinates for the hash mark circle are calculated and the clock is drawn.

The time to be drawn is specified by the ACLOCK structure member variables: Hour, Minute, and Second. The ranges of values allowed for these variables are as follows.

$0 \leq$ Hour ≤ 23

$0 \leq$ Minute ≤ 59

$0 \leq$ Second ≤ 59

The clock is not drawn correctly if a value exceeding one of these ranges has been set.

Two other functions are provided: DrawLine for drawing a line of the pixel thickness given by Width between two points (sx, sy) and (ex, ey); and DrawCircle for drawing a solid circle of pixels having center (x, y) and radius r. The prototypes for these functions are as follows.

```
void DrawLine( int sx, int sy, int ex, int ey, int Width );
void DrawCircle( int x, int y, int r );
```

[Program]

```
#include <math.h>
#define TRUE    1
#define FALSE   0

typedef  struct  {
  int    Ready;
  int    Hour, Minute, Second;
  int    Cx, Cy;
  int    Radius;
  int    PlateX[ 360 ], PlateY[ 360 ];
} ACLOCK;
```

```
void    DrawLine( int sx, int sy, int ex, int ey, int Width );
void    DrawCircle( int x, int y, int r );

void    DisplayClock( ACLOCK *pt )
{
 const  double Rad      = 3.14159265359 / 180.0;
 const  int  LargeMark  = 6;
 const  int  SmallMark  = 2;
 const  int  HourHand   = 7;
 const  int  MinuteHand = 3;
 const  int  SecondHand = 1;
 int    angle, Size, Hx, Hy, w, x, y;

 if ( pt->Ready == FALSE ) {    /* Calculate coordinates. */
      for ( w = -90, x = y = 0; w < 270; [ a ] ) {
          pt->PlateX[x] = pt->Cx + (int)(pt->Radius*cos(w*Rad));
          pt->PlateY[y] = pt->Cy + (int)(pt->Radius*sin(w*Rad));
      }
      pt->Ready = TRUE;
 }

 for ( angle = 0; angle < 360; angle += 6 ){ /* Draw hash marks. */
      if ( [ b ] )    Size = LargeMark;
      else            Size = SmallMark;
      DrawCircle( pt->PlateX[angle], pt->PlateY[angle], Size);
 }

 angle = pt->Second*6;                  /* Draw second hand. */
 DrawLine( pt->Cx, pt->Cy,
           pt->PlateX[angle], pt->PlateY[angle], SecondHand );

 angle = [ c ];                         /* Draw minute hand. */
 DrawLine( pt->Cx, pt->Cy,
           pt->PlateX[angle], pt->PlateY[angle], MinuteHand );

 if ( pt->Hour >= 12 ) pt->Hour -= 12;
 angle = [ d ];                         /* Draw hour hand. */
 Hx = ( pt->Cx + pt->PlateX[angle] ) / 2;
 Hx = ( Hx + [ e ] ) / 2;
 Hy = ( pt->Cy + pt->PlateY[angle] ) / 2;
 Hy = ( Hy + pt->PlateY[angle] ) / 2;
 DrawLine( pt->Cx, pt->Cy, Hx, Hy, HourHand );
 DrawCircle( pt->Cx, pt->Cy, pt->Radius / 10 ); /* Draw center axis. * /
}
```

Sub-Question

Select the correct answers from the answer groups below to be inserted into the corresponding boxes ( ) in the above program code.

Answer group for a:

a) `w++`
b) `w++, x++, y++`
c) `w++, x++, y--`
d) `w++, x+=6, y+=6`
e) `w+=6, x++, y++`
f) `w+=6, x+=6, y+=6`

Answer group for b:

a) `angle % 30 >= 0`
b) `angle % 30 != 0`
c) `angle % 30 <= 30`
d) `angle % 30 == 0`
e) `angle / 30 == 0`
f) `angle / 30 != 0`

C

Answer group for c:

a) `pt->Minute`
b) `pt->Minute*6`
c) `pt->Minute*6 + pt->Second`
d) `pt->Minute*6 + pt->Second/10`
e) `pt->Minute*6 + pt->Second/60`

Answer group for d:

a) `pt->Hour`
b) `pt->Hour*30`
c) `pt->Hour + pt->Minute`
d) `pt->Hour + pt->Minute/2`
e) `pt->Hour*30 + pt->Minute/2`
f) `pt->Hour*30 + pt->Minute/60`

Answer group for e:

a) `pt->Cx`
b) `pt->Cy`
c) `pt->PlateX[angle]`
d) `pt->PlateY[angle]`
e) `pt->PlateX[pt->Hour]`
f) `pt->PlateY[pt->Hour]`

Q7. Read the explanation of the following COBOL program then and answer the sub-question.

[Description of Program]

This sub-program searches through a human resource file for people having the required skills and displays the results found.

(1) Data representing human resource abilities classified into 10 skills are stored in the human resource file. Each skill is evaluated by ranking into one of four levels (3: High level, 2: Mid-level , 1: Novice, 0: None).

Record format used in the human resource file

Registration number 6 digits	Name 10 characters	Skill 1 1 digit	Skill 2 1 digit	≈	Skill 10 1 digit

The ranking for each skill is stored in Skill 1 through Skill 10.

(2) The required skills (used as a search condition) are passed as argument to the sub-program. A value is supplied for all 10 skills.

Argument format for search conditions

Condition 1 1 digit	Condition 2 1 digit	Condition 3 1 digit	Condition 4 1 digit	Condition 5 1 digit	Condition 6 1 digit	Condition 7 1 digit	Condition 8 1 digit	Condition 9 1 digit	Condition 10 1 digit

The required skill levels for Skills 1 through 10 in the human resource file (3: High level, 2: Mid-level or higher, 1: Novice or higher, 0: None) are stored in Conditions 1 through 10. All conditions must not be 0.

(3) People who have all the required skills are displayed in order beginning at those with the highest total ranking. The phrase “people having the required skills” means people who have a skill level at least equal to that required; for instance, people having a “high level” or “mid-level” of skill when “mid-level or higher” is passed as the condition argument for the required skill.

COBOL

[Program]

```
DATA              DIVISION.
FILE              SECTION.
FD  HRSRC-FILE.
01  HRSRC-REC.
    03  FILLER          PIC X(16).
    03  J-SKILL-INFO.
        05  J-SKILL     PIC 9(01)  OCCURS 10.
SD  HIT-FILE.
01  HIT-REC.
    03  G-POINT         PIC 9(02).
    03  G-HRSRC         PIC X(26).
WORKING-STORAGE   SECTION.
01  EOF-FLAG            PIC X(01)  VALUE SPACE.
01  MATCH               PIC 9(01).
01  K                   PIC 9(02).
LINKAGE           SECTION.
01  SEARCH.
    03  JOKEN           PIC 9(01)  OCCURS 10.
PROCEDURE         DIVISION  USING SEARCH.
MAIN-PROCESS.
    SORT HIT-FILE
         [ a ]
         INPUT PROCEDURE SEARCH-PROCESS
         OUTPUT PROCEDURE DISPLAY-PROCESS.
    EXIT PROGRAM.
SEARCH-PROCESS.
OPEN INPUT HRSRC-FILE.
READ HRSRC-FILE AT END MOVE "E" TO EOF-FLAG END-READ.
PERFORM UNTIL EOF-FLAG = "E"
   MOVE 0 TO G-POINT
   [ b ]
   PERFORM VARYING K FROM 1 BY 1 UNTIL K > 10 OR MATCH = 0
      IF [ c ]
         MOVE 0 TO MATCH
      ELSE
         IF [ d ]
            [ e ]
         END-IF
      END-IF
   END-PERFORM
   IF MATCH = 1
      MOVE HRSRC-REC TO G-HRSRC
      RELEASE  HIT-REC
   END-IF
   READ HRSRC-FILE AT END MOVE "E" TO EOF-FLAG END-READ
END-PERFORM.
CLOSE HRSRC-FILE.
```

COBOL

```
DISPLAY-PROCESS.
    MOVE SPACE TO EOF-FLAG.
    RETURN HIT-FILE AT END MOVE "E" TO EOF-FLAG END-RETURN.
    PERFORM UNTIL EOF-FLAG = "E"
       DISPLAY G-HRSRC
       RETURN HIT-FILE AT END MOVE "E" TO EOF-FLAG END-RETURN
    END-PERFORM.
```

Sub-Question

Select the correct answers from the answer groups below to be inserted into the corresponding boxes (☐) in the above program code.

Answer group for a:

a) `OCCURS KEY G-POINT`
b) `ON ASCENDING KEY G-POINT`
c) `ON DESCENDING KEY G-POINT`
d) `USING HRSRC-FILE`
e) `USING KEY G-POINT`

Answer group for b, e:

a) `COMPUTE G-POINT = G-POINT + J-SKILL(K)`
b) `COMPUTE G-POINT = G-POINT + COND(K)`
c) `INITIALIZE HIT-REC`
d) `MOVE 0 TO G-POINT`
e) `MOVE 0 TO MATCH`
f) `MOVE 1 TO MATCH`
g) `MOVE J-SKILL(K) TO G-POINT`
h) `MOVE COND(K) TO G-POINT`
i) `MOVE ZERO TO HIT-REC`

Answer group for c, d:

a) `J-SKILL(K) = 0`
b) `J-SKILL(K) > 0`
c) `J-SKILL(K) >= 0`
d) `J-SKILL(K) < COND(K)`
e) `J-SKILL(K) = COND(K)`
f) `J-SKILL(K) > COND(K)`
g) `COND(K) = 0`
h) `COND(K) > 0`
i) `COND(K) >= 0`

Q8. Read the explanation of the following assembler program and then answer the sub-question.

[Description of Program]
This program reads data from an input device, and outputs the number of words included in that data for each message.

(1) A message may be an empty message that includes no words at all, a single word, or multiple words each separated by one or more spaces. The end of a message is indicated by a " ." (period).
(2) A word is a string of characters consisting of other than spaces or periods.
(3) Input data is read in units of records. Although a single message may include words that cross multiple records, multiple messages may also be included in a single record. It is also possible that a record may not include any characters at all.
(4) The sub-program, `DECPRN`, used inside this program converts binary data passed by GR1 to decimal and outputs the result.

[Program]

```
WCNT   START
       LEA   GR2,0          ;Initialize input pointer
       ST    GR2,INLENG
LP1    LEA   GR1,0          ;Initialize number of words counter
LP2    CALL  GETCHR         ;Get a character
       JZE   FIN
       CPL   GR0,WDELM      ;Is it a word delimiter?
             a
       CPL   GR0,SDELM      ;Is it a sentence delimiter?
       JZE   SEND
LP3    CALL  GETCHR
       JZE   FIN
       CPL   GR0,WDELM
       JZE   WEND
       CPL   GR0,SDELM
       JNZ   LP3            ;If the character is not a delimiter, go to LP3
WEND   LEA   GR1,1,GR1      ;Increment the word count
       CPL   GR0,SDELM
             b
SEND   CALL  DECPRN         ;Output the number of words
       JMP   LP1
FIN    EXIT

GETCHR CPA   GR2,INLENG     ;Get a character
       JZE   GETREC
       LD    GR0,INBUFF,GR2
             c
       RET

GETREC LEA   GR2,0
       IN    INBUFF,INLENG  ;Read one record from the input device
             d
       CPA   GR0,EOF        ;Is this the end of input data?
       JNZ   GETCHR
       RET

WDELM  DC    ' '
SDELM  DC    '.'
EOF    DC    -1
INBUFF DS    80
INLENG DS    1
       END
```

Assembler

Sub-Question

Select the correct answers from the answer groups below to be inserted into the corresponding boxes (☐) in the above program.

Answer group for a:

a) `JNZ  LP2`
b) `JNZ  LP3`
c) `JZE  LP2`
d) `JZE  LP3`
c) `JZE  WEND`

Answer group for b:

a) `JNZ  LP2`
b) `JNZ  LP3`
c) `JZE  LP2`
d) `JZE  LP3`
c) `JNZ  WEND`

Answer group for c:

a) `LEA  GR1,1,GR1`
b) `LEA  GR2,-1,GR2`
c) `LEA  GR2,1,GR2`
d) `ST   GR0,INLENG`
e) `ST   GR2,INLENG`

Answer group for d:

a) `LD   GR0,0,GR2`
b) `LD   GR0,INBUFF,GR2`
c) `LD   GR0,INBUFF`
d) `LD   GR0,INLENG`

Select one question from Question 9 through Question 11, and mark [ ] which question you have selected on your answer sheet.
If you select more than one question, only the first selected question will be graded.

Q9. Read the explanation of the following C program and then answer Sub-Questions 1 through 3.

[Description of Program]
The function `decmult` is used to multiply two decimal numbers given as character strings.

(1) The arguments of the `decmult` function are `decA[]`, a character type array used to represent the multiplicand, `decB[]`, a character type array used to represent the multiplier, and `decC[]`, a character type array used to store the product.
(2) `decA[0]` and `decB[0]` may be used to store the characters "+" or "–" to represent the sign.
(3) The function `decmult` performs calculations using an algorithm similar to calculation with figures written down on paper.

Example:

	0	1	2	3	4	5
decA	1	2	3	4	5	\0

X

	0	1	2	3	4
decB	6	7	8	9	\0

=

	0	1	2	3	4	5	6	7	8
decC	8	3	8	1	0	2	0	5	\0

[Calculation by Writing]

			1	2	3	4	5		
		X		6	7	8	9		
		1	1^{2}	1^{3}	1^{4}	0^{4}	5		
		9^{1}	8^{2}	7^{3}	6^{4}	0			
	8^{1}	6^{2}	4^{3}	1^{3}	5				
7^{1}	4^{2}	0^{2}	7^{3}	0					
8^{1}	3^{1}	8^{2}	1^{1}	0^{1}	2	0	5		

[Program]

(Line No.)

```
01 #include <string.h>
02 #define  STRMAX  128
03
04 void decmult(char decA[], char decB[], char decC[])
05 {
06     char decT[STRMAX], sign='+';
07     int ai, bi, ci=0, ti=0, al=0, bl=0, am, bm, tm;
08     int carry=0, shift=0, wk;
09
10     memset(decT, '\0', STRMAX);  /* decT[]is initialized using '\0'*/
11     am = strlen(decA);
12     bm = strlen(decB);
13
14     /*** check sign ***/
15     if (decA[0] == '+' || decA[0] == '-')       al = 1;
16     if (decB[0] == '+' || decB[0] == '-')       bl = 1;
17     if ((decA[0] == '-' && decB[0] != '-') ||
18         (decA[0] != '-' && decB[0] == '-'))     sign = '-';
19
20     /*** multiply ***/
21     for (bi = bm - 1; bi >= bl; bi--) {
22         for (ai = am - 1; ai >= al; ai--, ti++) {
23             wk = (decA[ai] - '0') * (decB[bi] - '0') + carry;
24             carry = wk / 10;
25             wk %= 10;
26             if (decT[ti] != '\0') {  /* Is there a result for the previous operation? */
27                 wk += (decT[ti] - '0');
28                 if (wk > 9) {
29                     carry++;
30                     wk %= 10;
31                 }
32             }
33             decT[ti] = wk + '0';
34         }
35         if (carry > 0) {
36             decT[ti] = carry + '0';
37             carry = 0;
38         }
39         shift++;
40         ti = shift;
41     }
42
43     /*** store in decC[]***/
44     if (sign == '-') {
45         decC[0] = sign;
46         ci++;
47     }
48     tm = strlen(decT);
49     for (ti = tm - 1; ti > 0; ti--)
50         if (decT[ti] != '0')
51             break;
52     for ( ; ti >= 0; ti--, ci++)
53         decC[ci] = decT[ti];
54     decC[ci] = '\0';
55 }
```

C

C

Sub-Question 1

Select the correct pair of results when multiplications ① and ② below are performed. Note that "Δ" represents a character space.

	①	②
decA	−123	+123
decB	−4	−04

Answer group:

	①	②
a)	Δ492	−0492
b)	Δ492	−492
c)	492	−0492
d)	492	−492
e)	+492	−0492
f)	+492	−492

Sub-Question 2

Select the correct answer from the answer group below to be inserted into the following box (☐).

If multiplication of the multiplicand “234” and the multiplier “56” is performed, the 29th line of the program will be executed ☐ times.

Answer group:

a) 0 b) 1 c) 2 d) 3

e) 4 f) 5

Sub-Question 3

Revise the program so that it can handle multipliers that use decimal points. Select the correct answers from the answer groups below to be inserted into the corresponding boxes (☐) in the following table that gives the revised code.

C

Example:

	0	1	2	3	4	5	6	7
decA	0	.	1	2	3	\0		
			X					
	0	1	2	3	4			
decB	0	.	5	6	\0			
			=					
	0	1	2	3	4	5	6	7
decC	0	.	0	6	8	8	8	\0

Table Revised Code

Location	What to do	Code
Immediately before line 9	Add	`int ap, bp, tp;`
Line 11 and 12	Replace	`am = ap = strlen(decA);` `bm = bp = strlen(decB);`

Immediately before line 22	Add	`if (decB[bi] == '.') {`
		`  bp = bi + 1;`
		`  [ a ];`
		`}`

Immediately before line 23	Add	`if (decA[ai] == '.') {`
		`  ap = ai + 1;`
		`  [ b ];`
		`}`

Lines 49 through 54	Replace	`tp = [ c ];`
		`for (ti = tm - 1; [ d ]; ti--)`
		`  if (decT[ti] != '0')`
		`  break;`
		`for ( ; ti >= 0; ti--, ci++) {`
		`  if (ti == tp)`
		`  decC[ci++] = '.';`
		`  decC[ci] = decT[ti];`
		`}`
		`decC[ci] = '\0';`

Answer group for a and b:

a) `ai++` b) `ai--` c) `am++` d) `am--`
e) `bi++` f) `bi--` g) `bm++` h) `bm--`

Answer group for c:

a) `(am – ap) + (bm – bp)` b) `(am – ap) + (bm - bp) + 1`
c) `(am – ap) + (bm – bp) - 1` d) `ap + bp`
e) `ap + bp + 1` f) `ap + bp – 1`

Answer group for d:

a) `ti > 0` b) `ti > tp` c) `ti > tp + 1`
d) `ti >= 0` e) `ti >= tp` f) `ti >= tp – 1`

Q10. Read the explanation of the following COBOL program and then answer Sub-Questions 1 and 2.

[Description of Program]

This program creates a sales volume table from the monthly sales records of a company.

(1) The record format of the sales record file `SALES-F` is as follows.

Date 4 digits	Day of week code 1 digit	Store code 1 digit	Classification code 1 digit	Quantity 4 digits	Miscellaneous 20 digits

① Business days are from Monday through Saturday and a value from 1 to 6 is recorded in the day of week code as follows.
Monday: 1, Tuesday: 2, Wednesday: 3, Thursday: 4, Friday: 5, Saturday: 6

② With three stores, a value from 1 to 3 is recorded in the store code to represent the store number.

③ The products handled are classified into the following five categories and a value from 1 to 5 is recorded in the classification code as follows:
`Processed: 1, Vegetables: 2, Fruits: 3, Meat: 4, Fish: 5`

(2) The quantity of total sales of each product by day of week (up to 6 digits) is found and output in a table like the following.

```
All Stores
                      MON         TUE         WED         THU         FRI         SAT
Processed         ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9
Vegetables        ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9
Fruits            ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9
Meat              ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9
Fish              ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9     ZZZ,ZZ9
```

COBOL

[Program]

(Line No.)

```
01 DATA DIVISION.
02 FILE SECTION.
03 FD  SALES-F.
04 01  SALES -R.
05     03           PIC X(4).
06     03  S-WDAY   PIC 9(1).
07     03  S-STORE  PIC 9(1).
08     03  S-CATEG  PIC 9(1).
09     03  S-QTY    PIC 9(4).
10     03           PIC X(20).
11 WORKING-STORAGE SECTION.
12 01  TABLE VALUE ZERO.
13     03  T-CATEG  OCCURS 5 INDEXED BY H1.
14         05  T-WDAY  OCCURS 6 INDEXED BY H2 PIC 9(6).
15 01  END-SW    PIC X(3) VALUE SPACE.
16 01  HEADING1  PIC X(5).
17 01  HEADING2.
18     03        PIC X(15) VALUE SPACE.
19     03        PIC X(53) VALUE
20      "MON       TUE       WED       THU       FRI       SAT".
21 01  CATEG-TBL VALUE
22     "PROCESSED VEGETABLESFRUITS    MEAT     FISH".
23     03  M-CATEG OCCURS 5 INDEXED BY B1 PIC X(10).
24 01  DETAIL VALUE SPACE.
25     03  D-COLMN OCCURS 6 INDEXED BY M1.
26         05         PIC X(3).
27         05  D-QTY  PIC ZZZ,ZZ9.
28 PROCEDURE DIVISION.
29 START.
30     OPEN INPUT SALES-F.
31     READ SALES-F AT END MOVE "END" TO END-SW.
32     PERFORM  UNTIL END-SW = "END"
33        SET H1 TO S-CATEG
34        SET H2 TO S-WDAY
35        COMPUTE H-WDAY(H1 H2) =  T-WDAY(H1 H2) + S-QTY
36        READ SALES -F AT END MOVE "END" TO END-SW END-READ
37     END-PERFORM.
38     MOVE "ALL STORES" TO HEADING1.
39     DISPLAY  HEADING.
40     DISPLAY  HEADING2.
41     SET M1 B1 TO 1.
42     PERFORM  OUT-PROCESS VARYING H1 FROM 1 BY 1 UNTIL H1 > 5
43                          AFTER   H2 FROM 1 BY 1 UNTIL H2 > 6.
44     CLOSE  SALES-F.
45     STOP RUN.
46 OUT-PROCESS.
47     MOVE T-WDAY(H1 H2) TO D-QTY(M1).
48     SET M1 UP BY 1.
49     IF H2 = 6
50        DISPLAY D-CATEG(B1), DETAIL
51        MOVE SPACE TO DETAIL
52        SET M1 TO 1
53        SET B1 UP BY 1
54     END-IF.
```

Sub-Question 1

Although a different index is used for each table, it is possible to access multiple tables using shared variables if a subscript is used. If there are multiple tables in the program, the program can be made simpler by using subscripts instead of indices when it comes to accessing the same location. So it was decided to replace the indices H1, H2, B1 and M1 in the program with subscripts. Select the correct answer giving the combination of indices that can be replaced using shared subscripts.

Answer group:

a) H1 and B1
b) H1 and M1
c) H2 and B1
d) H2 and M1
e) H1 and B1, H2 and M1
f) H1 and M1, H2 and B1

Sub-Question 2

Revise the program so that it is possible to specify a day of the week to see what was sold where and when and output the total sales volume by store in the following format.

Day of Week 9

	Store 1	Store 2	Store 3
Processed foods	ZZZ,ZZ9	ZZZ,ZZ9	ZZZ,ZZ9
Vegetables	ZZZ,ZZ9	ZZZ,ZZ9	ZZZ,ZZ9
Fruits	ZZZ,ZZ9	ZZZ,ZZ9	ZZZ,ZZ9
Meat	ZZZ,ZZ9	ZZZ,ZZ9	ZZZ,ZZ9
Fish	ZZZ,ZZ9	ZZZ,ZZ9	ZZZ,ZZ9

Select the correct answers from the answer groups below to be inserted into the corresponding boxes (☐) in the following table that gives the revised code.

Table　Revised Code for Adding Functions

What to do	Code
Add after line 12	`02 T-STORE OCCURS 3 INDEXED BY H0.`
Add after line 15	a

Replace lines 18 through 20	`03 PIC X(12) VALUE SPACE.`
	`03 PIC X(26) VALUE`
	`" STORE 1  STORE 2  STORE 3".`

Replace line 35	`SET H0 TO S-STORE` `COMPUTE` b `=` b `+ S-QTY`

Replace lines 38 through 43	`MOVE "DAY OF THE WEEK" TO HEADING1`
	`PERFORM UNTIL SPEC-X = "Q"`
	`DISPLAY  "SPECIFIED DAY OF WEEK "`
	`ACCEPT SPEC-X`
	`EVALUATE SPEC-X`
	`WHEN "1" THRU "6"`
	`DISPLAY  HEADING1, SPEC-X`
	`DISPLAY  HEADING2`
	`SET H2 TO SPEC-9`
	`SET M1 B1 TO 1`
	c
	`WHEN "Q" CONTINUE`
	`WHEN OTHER DISPLAY "CORRECT DAY`
	`OF WEEK"`
	`END-EVALUATE`
	`END-PERFORM.`

Replace line 47	`MOVE` b `TO T-QTY (M1).`
Replace line 49	`IF H0 = 3`

Answer group for a:

a).
```
01 SPEC.
   02 SPEC-X PIC X VALUE ZERO.
   02 SPEC-9 PIC 9 VALUE 0.
```

b)
```
01 SPEC-X PIC X VALUE ZERO.
01 SPEC-9 PIC 9 VALUE 0.
```

c)
```
01 SPEC-X PIC X VALUE ZERO.
01 SPEC-9 REDEFINES SPEC-X PIC 9.
```

d)
```
01 SPEC-9.
   02 SPEC-X PIC X VALUE ZERO.
```

Answer group for b:

a) `T-WDAY(H0 H1 H2)`

b) `T-WDAY(H0 H2 H1)`

c) `T-WDAY(H1 H0 H2)`

d) `T-WDAY(H1 H2 H0)`

Answer group for c:

a)
```
PERFORM  OUT-PROCESS
     VARYING H0 FROM 1 BY 1 UNTIL H0 > 3
     AFTER   H1 FROM 1 BY 1 UNTIL H1 > 5
```

b)
```
PERFORM  OUT-PROCESS
     VARYING H0 FROM 1 BY 1 UNTIL H0 > 3
     AFTER   H2 FROM 1 BY 1 UNTIL H2 > 6
```

c)
```
PERFORM  OUT-PROCESS
     VARYING H1 FROM 1 BY 1 UNTIL H1 > 5
     AFTER   H0 FROM 1 BY 1 UNTIL H0 > 3
```

d)
```
PERFORM  OUT-PROCESS
     VARYING H1 FROM 1 BY 1 UNTIL H1 > 5
     AFTER   H2 FROM 1 BY 1 UNTIL H2 > 6
```

e)
```
PERFORM  OUT-PROCESS
     VARYING H2 FROM 1 BY 1 UNTIL H2 > 6
     AFTER   H0 FROM 1 BY 1 UNTIL H0 > 3
```

f)
```
PERFORM  OUT-PROCESS
     VARYING H2 FROM 1 BY 1 UNTIL H2 > 6
     AFTER   H1 FROM 1 BY 1 UNTIL H1 > 5
```

Q11. Read the explanation of the following assembler program and then answer Sub-Questions 1 through 4.

[Description of Program]

The sub-program `AFLD` inserts the bit pattern received into a word. The sub-program `MVFLD` for moving the bit pattern is used in creating `AFLD`.

(1) `MVFLD` moves bit pattern data as shown in Fig. 1.

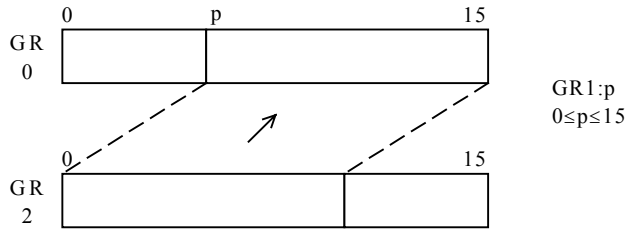


Fig. 1 Movement of Bit Pattern Data

(2) According to the parameter in the format shown in Figure 2, `AFLD` inserts bit data of length m starting at the 0th bit of source into the p-th bit of the target. The start address of the parameter is stored in GR1 and passed from the main program.

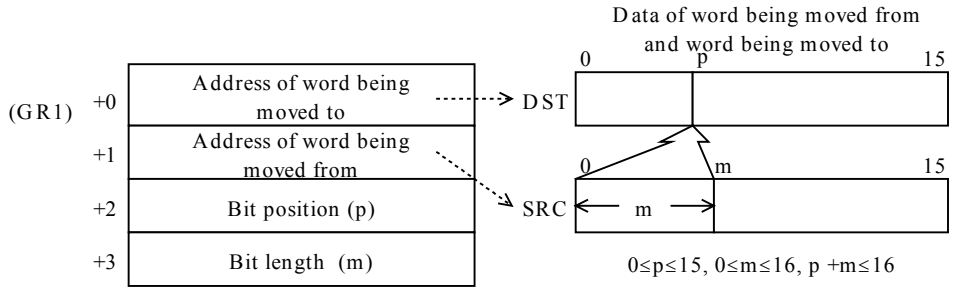


Fig. 2 Parameter Format

(3) When each subprogram returns to the program which called it, the data in the general purpose registers GR1 through GR3 is returned to its previous state.

[Program]

(Line No.)

```
01      MVFLD   PUSH  0,GR2          ; Back up the register
02              PUSH  0,GR3          ;
03              LD    GR3,FULLB      ; Create a mask
04              SRL   GR3,0,GR1      ;
05              EOR   GR3,FULLB      ;
06              ST    GR3,TEMP       ; Store mask data
07              AND   GR0,TEMP       ; Clear the copy-to area
08              SRL   GR2,0,GR1      ; Adjust the location of the data in the copy-from area
09              ST    GR2,TEMP       ;
10              OR    GR0,TEMP       ; Store data in the copy-from area
11              POP   GR3            ; Restore the register
12              POP   GR2            ;
13              RET                  ;
14      TEMP    DS    1              ;
15      FULLB   DC    #FFFF          ;
16      ;
17      AFLD    PUSH  0,GR2          ; Back up the register
18              PUSH  0,GR3          ;
19              LD    GR3,0,GR1      ; Get the contents of DST
20              LD    GR0,0,GR3      ;
21              ST    GR0,SVFLD      ; Temporarily save the contents of DST
22              LD    GR3,1,GR1      ; Get the contents of SRC
23              LD    GR2,0,GR3      ;
24              PUSH  0,GR1          ;
25              LD    GR1,2,GR1      ; Get the location of DST
26              CALL  MVFLD          ; Store the contents of SRC into DST
27              POP   GR1            ;
28              LD    GR2,SVFLD      ; Restore the contents of DST
29              LD    GR3,2,GR1      ; Find the number of SRC bits to be moved
30              SLL   GR2,0,GR3      ; Pad the contents of DST to the left
31              PUSH  0,GR1          ;
32              ADD   GR3,3,GR1      ; Find the location of DST
33              LEA   GR1,0,GR3      ;
34              CALL  MVFLD          ; Store the data into DST
35              POP   GR1            ;
36              LD    GR3,0,GR1      ;
37              ST    GR0,0,GR3      ; Store the result
38              POP   GR3            ; Restore the register
39              POP   GR2            ;
40              RET                  ;
41      SVFLD   DS    1              ;
42              END
```

Sub-Question 1

MVFLD is called using the register values shown below. Select the correct answer from the choices below for the value of TEMP immediately before the RET instruction is executed on line 13.

GR0	#1234
GR1	8
GR2	#5600

Answer group:

a) #0056 b) #00FF c) #1234 d) #1256 e) #FF00

Sub-Question 2

The program was changed as follows to add to MVFLD the process which does not require the use of bit pattern movement. Select the correct answer from the choices below to be inserted into the box ( ) in the program.

```
MVFLD [          ]
        JZE    ZEROP
PUSH  0,GR2          ; Back up register
   .
   .                                  Original
   .                                  processing
POP   GR2            ;
RET                  ;

ZEROP  LEA   GR0,0,GR2     ;
       RET
TEMP   DS    1             ;
FULLB  DC    #FFFF         ;
       .
       .
       .
```

Assembler

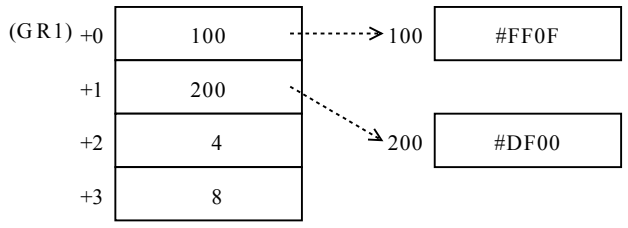
Answer group:

a) LD GR0,0,GR1 b) LD GR1,0,GR1 c) LD GR2,0,GR1

d) LEA GR1,0,GR1 e) LEA GR1,0,GR2 f) LEA GR1,0,GR3

Sub-Question 3

`AFLD` is called with the following argrements. Select the correct answer from the choices below for the value of GR2 immediately after executing the instruction on line 30 given on page 51.



Answer group:

a) `#00F0` b) `#F0F0` c) `#FF0F` d) `#FFF0` e) `#FFFF`

Sub-Question 4

When a bit pattern is inserted using `AFLD`, there is a bit pattern that is thrown away from the position of insertion. A program for storing the thrown-away bit pattern in GR0, padded to the left, was added immediately after line 37 on page 51. Select the correct answer from the choices below to be inserted into the following [ ] code to be added to the program.

```
LEA   GR3,16
SUB   GR3,2,GR1
SUB   GR3,3,GR1
      [          ]
LEA   GR0,0,GR2
```

Answer group:

a) `SLL GR2,0,GR1` b) `SLL GR2,0,GR3` c) `SLL GR3,0,GR2`
d) `SRL GR2,0,GR1` e) `SRL GR2,0,GR3` f) `SRL GR3,0,GR2`