



October 2009

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**

For (Q6, Q7) and (Q8, Q9), mark the Ⓢ of the question you select to answer in the “Selection Column” on your answer sheet.

(4) **Answers**

Mark your answers as shown in the following sample question.

[Sample Question]

In which month is the autumn Fundamental IT Engineer Examination conducted?

Answer group

a) September b) October c) November d) December

Since the correct answer is “b) October”, mark your answer sheet as follows:

[Sample Answer]

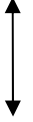


1	Ⓐ	●	Ⓒ	Ⓓ
---	---	---	---	---

Do not open the exam booklet until instructed to do so.


Inquiries about the exam questions will not be answered.

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

[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. The Process is executed while the Conditional expression is true.

[Operator]

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

[Logical type constant]

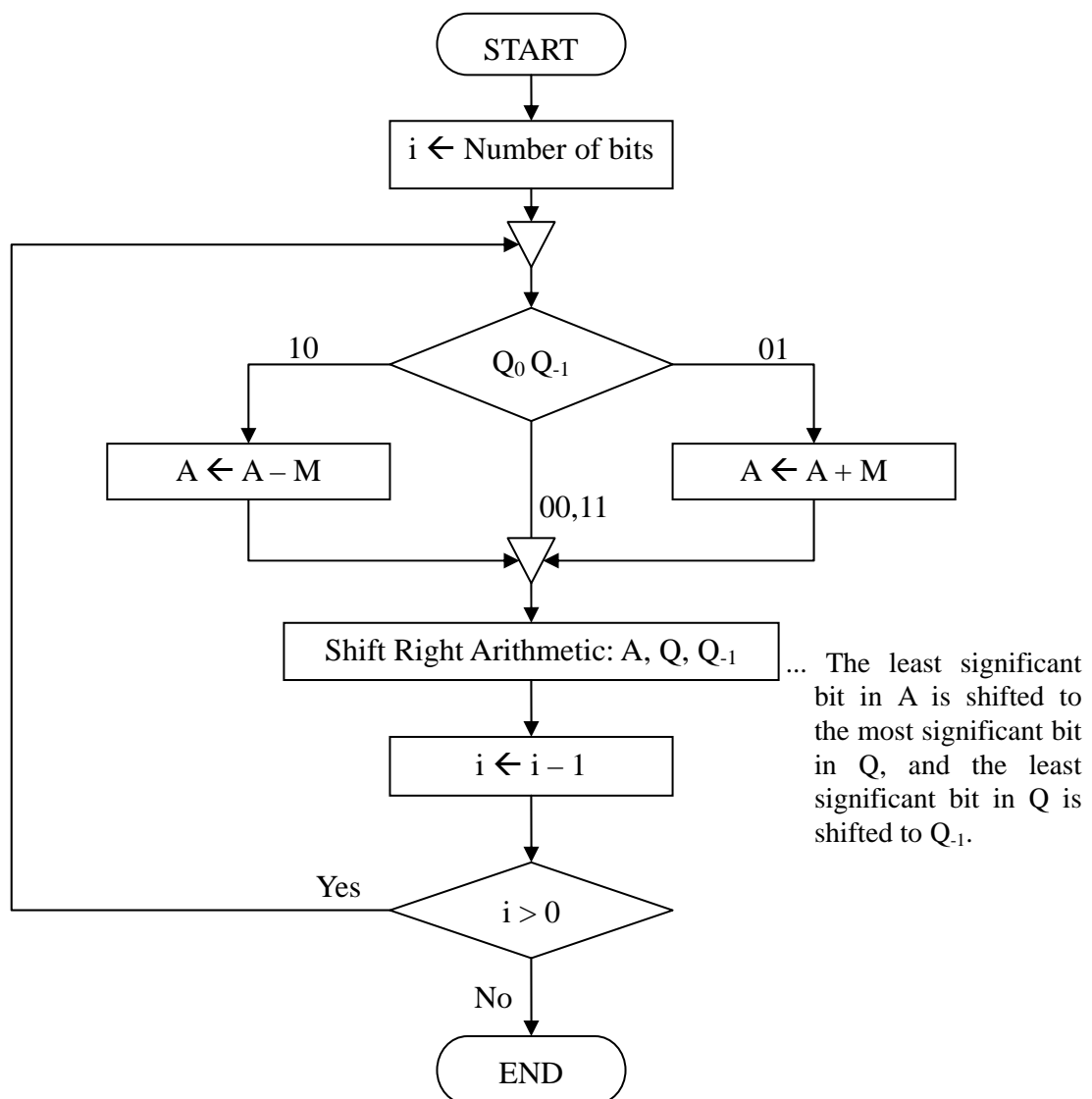
true false

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

Q1 Read the following description concerning multiplication operation on binary integers, and then answer Subquestion.

Generally, electronic circuits such as ALU's execute multiplication operation on signed binary integers by using addition, subtraction and shifting operations. Booth's algorithm shown in the flowchart below can execute multiplication effectively for this purpose.

There are three 4-bit registers: A, Q and M. Two signed binary integers, multiplicand and multiplier, are first stored in registers Q and M, respectively. In the flowchart below, Q_0 represents the least significant bit in register Q. Q_{-1} holds the bit that will be shifted out from Q by Shift right arithmetic operation. A and Q_{-1} are initialized to 0. The final product will be stored in registers A and Q as an 8-bit signed integer. Here, assume that $M \neq -8$.



The following figure illustrates the calculation of $5 (0101) \times -3 (1101) = -15 (1111\ 0001)$, using the Booth's algorithm.

A	Q	Q ₋₁	M	Operation	Note
0000	0101	0	1101		Initial values
0011	0101	0	1101	Add or Subtract	Cycle 1
0001	1010	1	1101	Arithmetic Shift	
<div style="border: 1px solid black; padding: 2px;">A</div>	1010	1	1101	Add or Subtract	Cycle 2
<div style="border: 1px solid black; padding: 2px;">B</div>	0101	0	1101	Arithmetic Shift	
<div style="border: 1px solid black; padding: 2px;">C</div>	0101	0	1101	Add or Subtract	Cycle 3
<div style="border: 1px solid black; padding: 2px;">D</div>	0010	1	1101	Arithmetic Shift	
1110	0010	1	1101	Add or Subtract	Cycle 4
1111	0001	0	1101	Arithmetic Shift	

Subquestion

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

in the above figure.

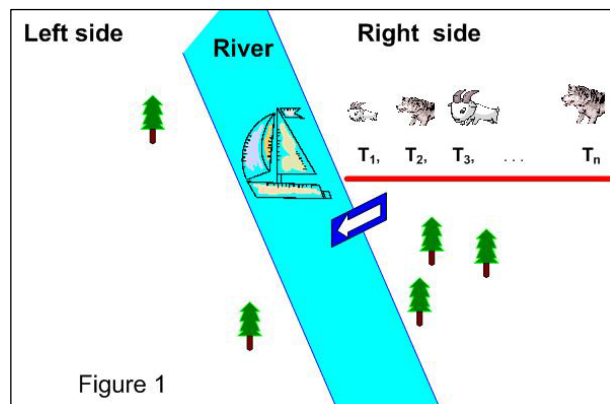
Answer group

- a) 0000
- b) 0001
- c) 0010
- d) 0011
- e) 1100
- f) 1101
- g) 1110
- h) 1111

Q2 Read the following description concerning the solution of a problem, and then answer Subquestion.

This is one of old Mongolian problems.
There is a land shown in Figure 1, which is separated by a river into **Right** and **Left** sides.

There are n animals, $T_1, T_2, T_3, \dots, T_n$, in the **Right** side. The provided condition is that animal T_k eats animal T_{k-1} but is eaten by T_{k+1} ($1 < k < n$), then move the animals to the **Left** side alive using a boat.



Here:

1. n is any positive integer number.
2. A boat can carry up to m animals, where $m = \lfloor n / 2 \rfloor$, an integer part of $n / 2$. The owner of animals sails a boat, and the owner's weight is not included in m .
3. Not let the animals eat each other while they are being moved from the **Right** side to the **Left** side of the river.
4. The animals won't eat each other when the owner is with them.
5. The owner will load and unload the animals as well as sail the boat until the process is finished.
6. It is allowed to take the animals, already in the **Left** side, back to the **Right** side.

[Sample solution steps for $n = 3$]

Input:

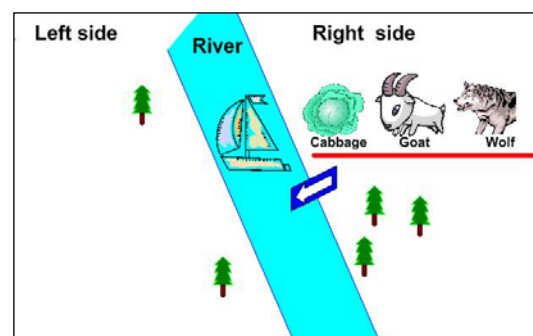
1. $n = 3$.
2. $T_1 = \text{Cabbage}$; $T_2 = \text{Goat}$; $T_3 = \text{Wolf}$.
3. $m = 1$ (the boat can carry one animal besides the owner).

Output:

1. **Cabbage**, **Goat** and **Wolf** should be taken to the **Left** side alive.

Process:

1. Take the **Goat** from **Right** side to **Left** side.
2. Go back from **Left** side to **Right** side alone.
3. Take the **Cabbage** from **Right** side to **Left** side.
4. Take back the **Goat** from **Left** side to **Right** side.



5. Take the **Wolf** from **Right** side to **Left** side.
6. Go back from **Left** side to **Right** side alone.
7. Take the **Goat** from **Right** side to **Left** side.
8. Stop.

[Solution steps for any positive integer n]

Input:

1. n : any positive integer number.
2. $T_1, T_2, T_3, \dots, T_n$: the animals that satisfy the given condition.
3. $m : \lfloor n / 2 \rfloor$ (the boat can carry m animals at once besides the owner).

Output:

1. n animals, $T_1, T_2, T_3, \dots, T_n$, should be taken alive to the **Left** side.

Process:

1. Take from the **Right** side A to the **Left** side.
2. Go back from **Left** side to **Right** side alone.
3. Take from the **Right** side B to the **Left** side.
4. If an animal is left in the **Right** side, then go to Step 5, else go to Step 9.
5. Take from the **Left** side C to the **Right** side.
6. Take from the **Right** side D to the **Left** side.
7. Go back from **Left** side to **Right** side alone.
8. Take from the **Right** side E to the **Left** side.
9. Stop.

Subquestion

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

in the above description.

The same answer can be selected more than once, if needed.

Answer group

- a) T_1
- b) T_2
- c) T_{2^*m}
- d) T_{2^*m+1}
- e) $T_1, T_2, T_3, \dots, T_{m-1}, T_m$
- f) $T_{m+1}, T_{m+2}, T_{m+3}, \dots, T_{n-1}, T_n$
- g) $T_1, T_3, T_5, \dots, T_{2^*m-3}, T_{2^*m-1}$
- h) $T_2, T_4, T_6, \dots, T_{2^*(m-1)}, T_{2^*m}$

Q3 Read the following description concerning relational database design, and then answer Subquestions 1 through 3.

A hospital aims to build a database for patient admission and treatment.

The results from the requirement analysis are as follows:

The hospital has a large number of registered physicians. Information for a physician need to be recorded is physician name, physician address, contact phone, and specialty. Patients are admitted to the hospital by a physician. Patient information such as patient name, patient address, and contact phone, is recorded at the time of admission. Any patient who is admitted must have exactly one admitting physician. A physician may admit any number of patients. Once admitted, a patient must be treated by at least one physician. A particular physician may treat any number of patients, or may not treat any patients.

The following table shows explanation of symbols used in the question.

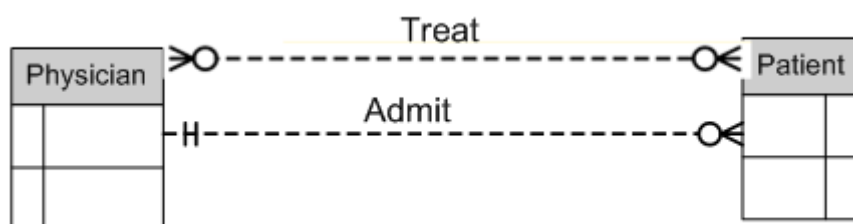
----->=	One or more
----->O	Zero or More
-H-----	Exactly One
+O-----	Zero or One

Subquestion 1

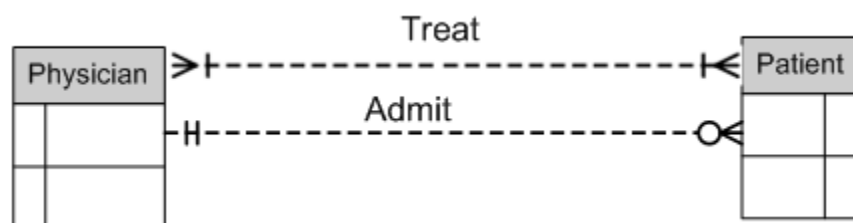
From the answer group below, select the correct Entity Relationship Diagram to fulfill the business requirement.

Answer group

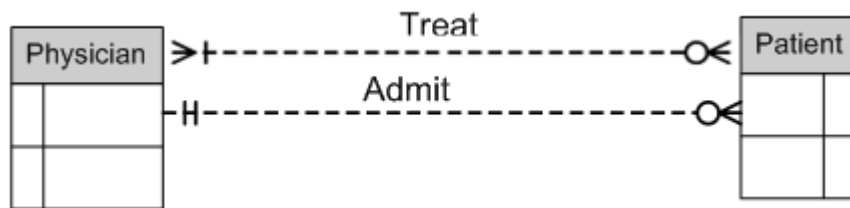
a)



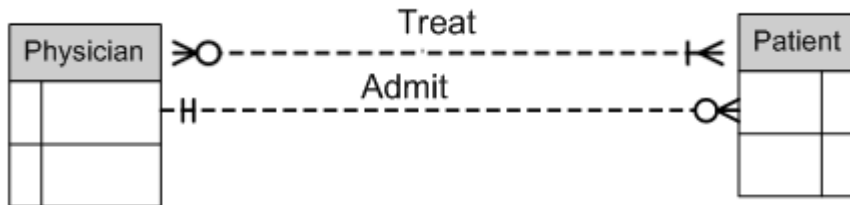
b)



c)



d)



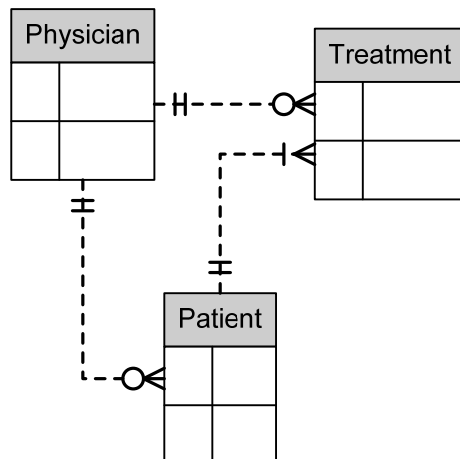
Subquestion 2

Many-to-Many relationship cannot be implemented for relational database management system so that it is required to do relational analysis to get finalized ERD (Entity Relationship Diagram).

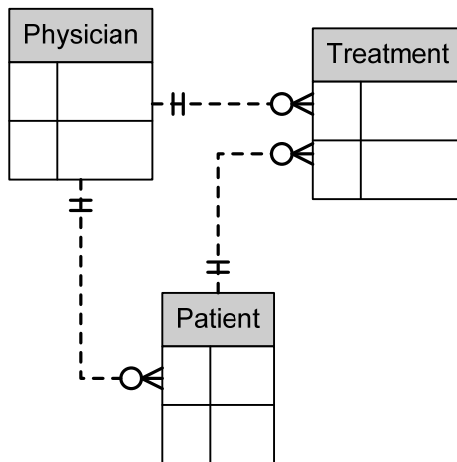
From the answer group below, select the correct ERD after performing relational analysis.

Answer group

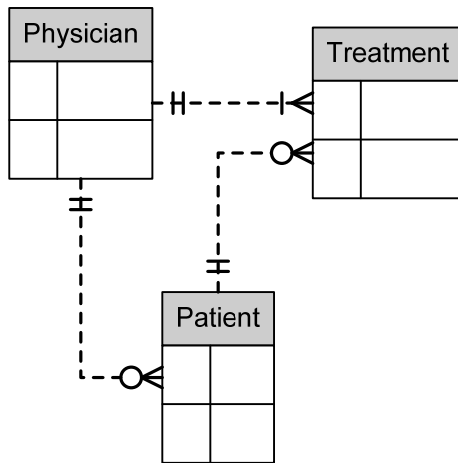
a)



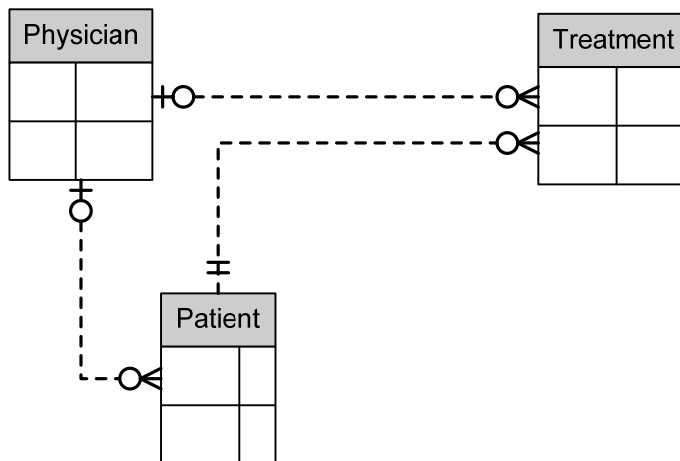
b)



c)



d)



Subquestion 3

The hospital requires two reports: Treatment history for particular patient including information of physicians, and Statement of daily admission report for a particular physician. For the sake of performance, which of the following is correct use of index

From the answer group below, select the correct use of indexes in order to improve the database performance.

Answer group

- a) [Index on Patient No for Patient table], [Index on Physician No for Physician table]
- b) [Index on Patient No for Patient table], [Index on Physician No for Physician table],
[Index on Physician No and Patient No for Treatment table]
- c) [Index on Physician No for Physician table],
[Index on Physician No for Treatment table]
- d) [Index on Patient No for Patient table], [Index on Patient No for Treatment table]

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

[Program Description]

`eightQueens` is a subprogram that puts eight chess queens on 8×8 chessboard. The queens must be placed in such a way that no two queens would be able to attack each other using the standard chess queen's moves. Figure 1-(1) illustrates the power of a single queen (indicated by a Q character).

Thus, a solution requires that no two queens share the same row, column, or diagonal line. Figure 1-(2) shows a sample solution to this problem.

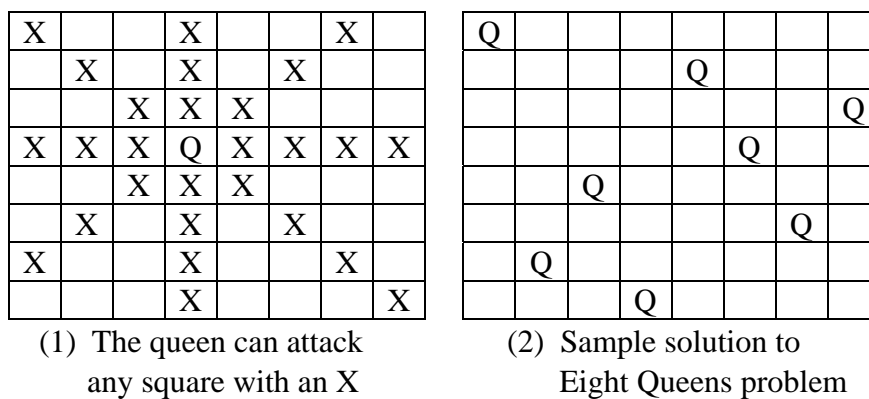


Figure 1. The Eight Queens Problem

- (1) The subprogram `eightQueens` is called recursively. It involves both recursive and interactive looping.
 - a. A recursive call is made after each successful placement of a queen, if an eighth queen has not been placed.
 - b. A return from recursive call indicates that one of two events has happened.
 - In the failure case, the algorithm failed to place the next queen and must therefore backtrack to the last-placed queen, remove it, place it at the next safe board position, and then look for the next safe board position for the next queen.
 - In the success case, the algorithm just placed all eight queens and must now unwind completely from all recursive calls without attempting to place other queens or disturbing already-placed queens.
- (2) Chess board is presented by using a two-dimensional character array `board[] []`. Indexes of rows and columns of this array is from 1 to 8 and the initial values of elements of this array are set to ' '.
- (3) A global variable `queenCount` is used to store the number of the safe-placed Queens.

The initial value of this variable is set to 0.

- (4) The initial call is `eightQueens(1,1)`.
- (5) `eightQueens` uses the subprogram `findSafePosition` that finds the next safe position starting with the current board position given in its parameters. If `findSafePosition` doesn't find a safe position, it returns a false value; otherwise it returns a true value as well as the row and column of the safe position through the parameters.
- (6) The subprogram `isAttack` checks if the position given in its parameters is the safe position or not. If this position is the safe position, it returns a true value; otherwise it returns a false value.
- (7) The parameters of the subprograms are shown in the tables below.

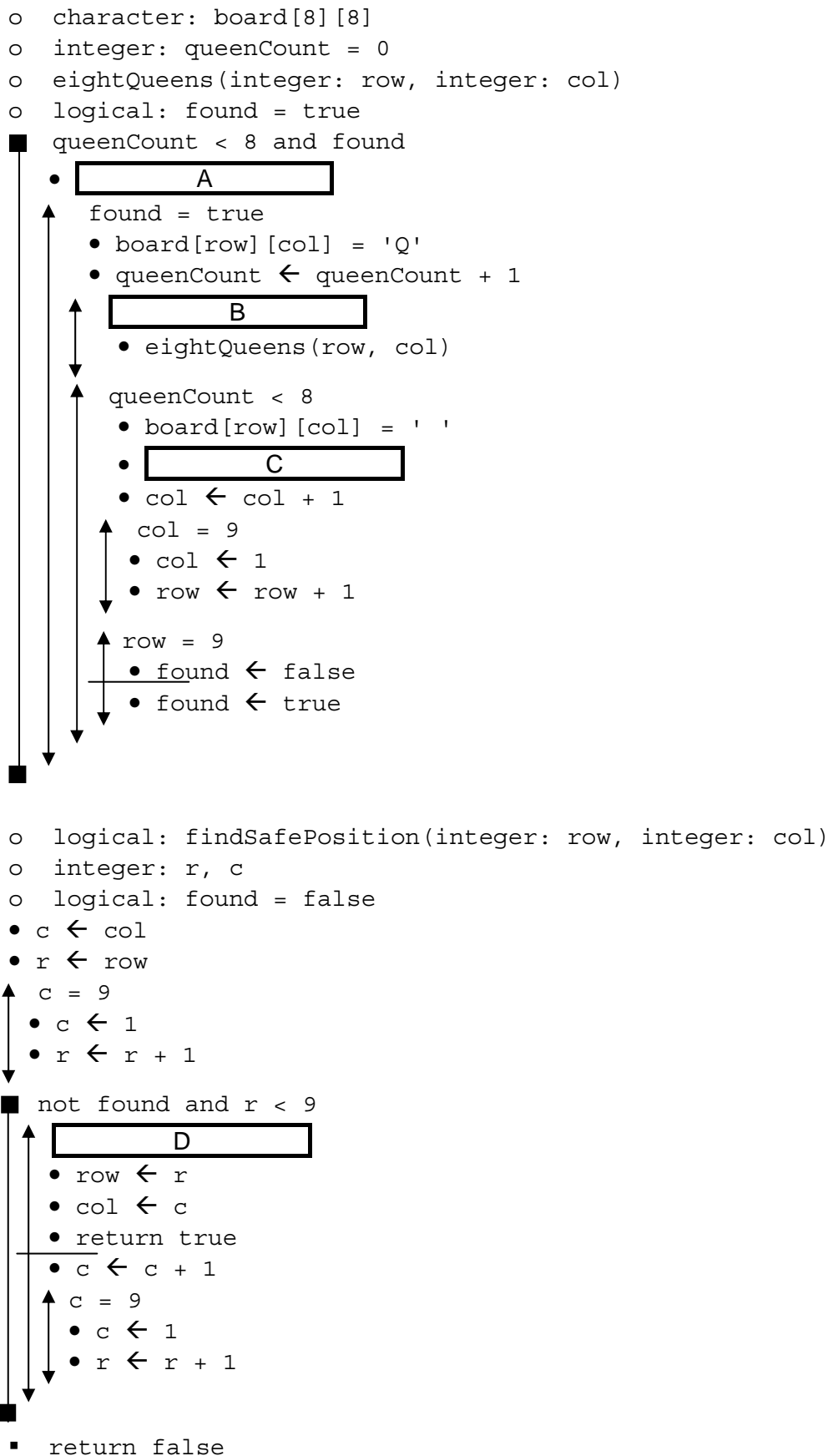
Table 1 Specification of the parameters of `findSafePosition`

Name of parameter	Input/Output	Description
Row	Input/Output	The row of the current position for searching the next safe position. It also is the row of the safe position returned if found
Col	Input/Output	The column of the current position for searching the next safe position. It also is the column of the safe position returned if found

Table 2 Specification of the parameters of `isAttack`

Name of parameter	Input/Output	Description
R	Input	The row of the position for checking
C	Input	The column of the position for checking

[Program]



```

o logical: isAttack(integer: r, integer: c)
o integer: r1, c1
• r1 ← 1
■ r1 ≤ r
  • c1 ← 1
  ■ c1 < 9
    ↑ board[r1][c1] = 'Q'
    ↑ E
    • return true
    ↓
  • c1 ← c1 + 1
  ■
  • r1 ← r1 + 1
■
• return false

```

Subquestion 1

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

Here, the function abs returns the absolute value of the integer parameter.

Answer group for A and C

- a) found ← findSafePosition(row, col + 1)
- b) found ← findSafePosition(row, col)
- c) queenCount ← queenCount - 1
- d) queenCount ← queenCount + 1

Answer group for B

- a) found = false
- b) queenCount < 8
- c) queenCount = 8
- d) queenCount > 8

Answer group for D

- a) isAttack(r, c) = false
- b) isAttack(r, c) = true
- c) isAttack(r1, c1) = false
- d) isAttack(r1, c1) = true

Answer group for E

- a) r = r1 and c = c1 and abs(r - r1) = abs(c - c1)
- b) r = r1 and c = c1 or abs(r - r1) = abs(c - c1)
- c) r = r1 or c = c1 and abs(r - r1) = abs(c - c1)
- d) r = r1 or c = c1 or abs(r - r1) = abs(c - c1)

Subquestion 2

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

After 5 times that the `eightQueens` subprogram is called, the 5th queen is placed at the position F . After 6 times, the 6th queen is placed at the position G . Here, a position is represented by “(row, column)”.

Answer group

- | | |
|-----------|-----------|
| a) (4, 2) | b) (5, 4) |
| c) (5, 8) | d) (7, 4) |

Q5 Read the following description concerning a project costing system, and then answer Subquestion.

A project costing system consists of three subsystems. The first subsystem is time management that records time of work of all employees. The second subsystem is payroll that makes use of data from the time management subsystem to calculate employees' weekly pay. The third subsystem is project monitoring that calculates computer usage and employees' pay related to each of the projects.

[File Description]

- (1) Employee file contains employee id, employee name, pay rate for regular hours, overtime rate for overtime hours, and employee's terminal id. An employee's total pay is the sum of regular hours multiplied by pay rate and overtime hours multiplied by overtime rate.

The record format for Employee file:

employee_id	employee_name	pay_rate	ot_rate	terminal_id
-------------	---------------	----------	---------	-------------

- (2) Project file contains project id, project name, and manager id.

The record format for Project file:

project_id	project_name	manager_id
------------	--------------	------------

- (3) Timelog file is maintained by the time management subsystem. It contains daily time records. Employees time-in and time-out each time they start and end work on a specific project. They can work on several projects in a day but never simultaneously. They also work on a project only once in a day. Timelog is only a log of an event. A record can only have either time-in or time-out but never both. The other field will be NULL in value.

The record format for Timelog file:

date	time_in	time_out	employee_id	project_id
------	---------	----------	-------------	------------

- (4) Mainframelog file is a system generated mainframe usage log. Mainframe use costs \$0.10 per second, and terminal use costs \$3 per hour.

The record format for Mainframelog file:

terminal_id	job_id	date	start_time	end_time	elapsed_time
-------------	--------	------	------------	----------	--------------

[Program Description]

- (1) Employees log on to the system with the employee id and the project id each time they start to work for a project. They get automatically logged out when they log in to a new project or shutdown to leave. Timelog record is created based on this activity.

- (2) Time management subsystem creates a DTS (daily time summary) file at the end of the day. It contains employee id, project id, worked hours per project, and overtime hours per project.

The record format for DTS file:

date	employee_id	project_id	worked_hours	ot_hours
------	-------------	------------	--------------	----------

- (3) Overtime is classified as time worked in excess of 8 hours each day. But if the total time for the day is less than 9 hours, it is still classified as regular work hours.
- (4) Payroll subsystem prepares a weekly summary and calculates the weekly pay. It generates a weekly payroll report sorted by employee name, as shown below.

Weekly Payroll Ending <date>				
Employee ID	Name	Worked Hours	OT Hours	Salary
9873	James Nguyen	48.5	8.0	\$525.00
7890	Peter Ng	36.0	2.0	\$370.00

- (5) At the end of each day, the actual overtime hours of every employee is distributed proportionately across the projects he/she worked on for that day.
- (6) Project managers monitor project costing based on the employee salary and also computer time (both mainframe and terminal use). They periodically generate a project costing report sorted by project name.

Project ID	Name	Time (min)		Cost	Salary	Total Cost
		mainframe	terminal			
984	Annealing Test	8.5	80.0	\$55.0	\$800	\$855.00
129	Genetic Run	16.0	245.0	\$108.25	\$2700	\$2808.25

- (7) ErrorRtn (record, message) is a procedure that writes to an Error log file. A record that is in error and the corresponding error message will be recorded.
- (8) Flowchart(1) in Figure 1 calculates the worked hours per project (overtime hours per project is not calculated in this flowchart), and outputs it to the DTS file.

For the execution of flowchart(1), the Timelog file must be sorted by

A

, each in ascending order.

- (9) After computing worked hours per project and creating DTS file by Flowchart(1) in Figure 1, a temporary file, which contains the total hours worked per day of each employee, is created from the DTS file. The temporary file is used along with the DTS file in Flowchart(2) in Figure 2, in order to compute overtime hours per project. The minimum fields needed by the temporary file are **D**, and total_hours. Here, the total_hours contains the total hours worked per day of each employee.

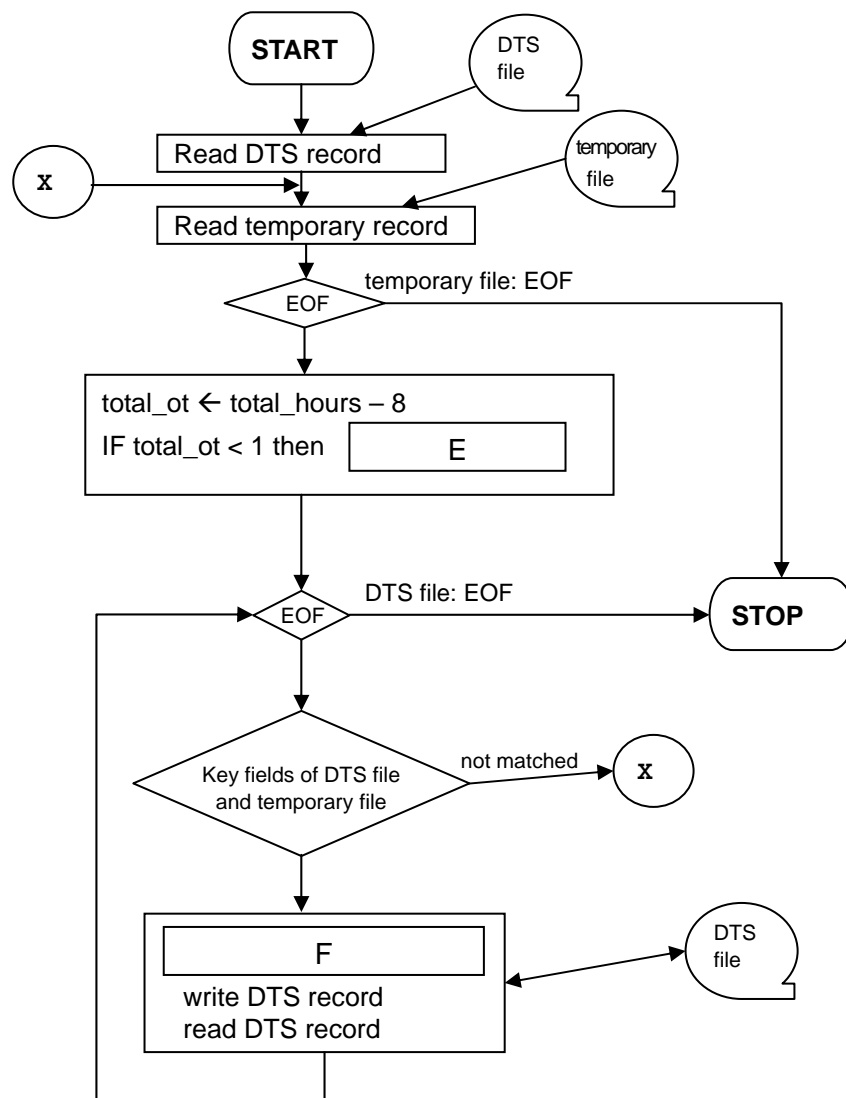


Figure 2. Flowchart(2)

- (10) The payroll subsystem needs to use the files **G** and **H** to generate the weekly payroll report. The project costing report will use all the files except the temporary file.

Subquestion

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

in the above description.

Answer group for A and D

- a) date, employee_id
- b) date, employee_id, project_id, time_in
- c) date, employee_id, project_id, time_out
- d) employee_id, project_id
- e) project_id, employee_id, time_in

Answer group for B and C

- a) employee ID or project ID did not match
- b) employee ID or project ID is empty
- c) employee ID or project ID was not found
- d) time in has null values
- e) time out has null values
- f) worked hours cannot be computed

Answer group for E and F

- a) $ot_hours \leftarrow 0$
- b) $ot_hours \leftarrow total_ot / project_count$
- c) $ot_hours \leftarrow worked_hours / total_hours$
- d) $ot_hours \leftarrow worked_hours / total_hours * total_ot$
- e) $total_ot \leftarrow 0$
- f) $total_ot \leftarrow 1$

Answer group for G and H

- a) DTS
- b) Employee
- c) Mainframelog
- d) Project
- e) Timelog

Concerning questions **Q6** and **Q7**, **select one** of the two questions. Then, mark **(S)** in the selection area on the answer sheet, and answer the question.

If **two questions** are selected, **only the first question** will be graded.

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

[Program Description]

The program searches a search string of length n for a pattern string of length m . Each time a pattern string is found in a search string, the program outputs its character position. A search string contains 27 types of characters: ' ' and 'A' - 'Z'.

The following figure illustrates how the program finds the pattern string "SING" in the search string "THIS SEARCH USING".

Character position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Search String	T	H	I	S		S	E	A	R	C	H		U	S	I	N	G
Pattern String	S	I	N	G													
1st comparison	S	I	N	G													
2nd comparison				S	I	N	G										
3rd comparison								S	I	N	G						
4th comparison												S	I	N	G		
5th comparison														S	I	N	G

1st comparison: Locate the pattern string at the top (character position 0) of the search string. Then, check if the first 4 characters of the search string match with "SING".

Comparison is done character by character, from right to left. If the characters are matched, then move to the previous position and repeat the same operation. Otherwise, examine the unmatched character in the search string (in this case, "S" at position 3) to check if it appears in the pattern string. In this case, "S" appears in the pattern string. Then, shift the pattern string to the right so that both "S"s share the same position.

2nd comparison: The same operation as in 1st comparison is done. In this comparison, the unmatched character in the search string ("E" at position 6) does not appear in the pattern string. Then, shift the pattern string m (in this case, 4) positions to the right.

3rd - 5th comparison: The same operations as in 1st and 2nd comparison are done. Finally, at position 13, the string "SING" is found, and the program outputs the following message.

Found at position 13

[Program]

```
#include <stdio.h>
// ASIZE represents the number of characters used (' ', 'A' - 'Z').
#define ASIZE 27

void OUTPUT (int a) {
    printf ("Found at position %d\n", a);
}

/*  Function: Maps a character to an index in the table.
    0 is assign to ' ', and 1 - 26 are assigned to 'A' - 'Z'.
*/
int getIndex (char x) {
    int ret_val;

    if (x == ' ')    // if the character is a space
        ret_val = 0;
    else
        ret_val = A ;
    return ret_val;
}

/*  Function: Determines slide positions of each character.
    *x - pattern string
    m - length of pattern string
    Slideto - array containing slide positions
*/
void computeSlideToPosition(char *x, int m, int slideTo []) {
    int i;

    for (i = 0; i < ASIZE; ++i)
        slideTo [i] = m;
    for (i = 0; i < m - 1; ++i)
        slideTo [getIndex(x[i])] = m - i - 1;
}
```



```

/*  Function: Search for pattern string *x in the search string *y.
    *x - pattern string
    m - length of pattern string
    *y - search string
    n - length of search string
*/
void searchString(char *x, int m, char *y, int n) {
    int i, j, incr, slideTo[ASIZE];

    computeSlideToPosition (x, m, slideTo);

    j = 0;
    while (j <= ) {
        for (i = m - 1; i >= 0 && x[i] == y[i + j]; --i);

        if (  ) {
            OUTPUT(j);
            j += 1;
        }
        else {
            incr = slideTo [getIndex(y[i + j])] - m  ;
            if (incr > 0) {
                j += incr;
            }
            else {
                j += 1;
            }
        }
    }
}

int main(void) {
    searchString("SING", 4, "THIS SEARCH USING", 17);
    return 0;
}

```

Subquestion

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

in the above program.

Answer group for A

- | | |
|----------------|----------------|
| a) x | b) x - 'A' |
| c) x - 'A' - 1 | d) x - 'A' + 1 |

Answer group for **B**

a) m

b) n

c) $n - m$

d) $n + m$

Answer group for **C**

a) $i < 0$

b) $i > 0$

c) $j < 0$

d) $j > 0$

Answer group for **D**

a) $+ 1$

b) $+ i$

c) $+ i - 1$

d) $+ i + 1$

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

[Program Description]

The program 1 illustrates *member* variables (class and instance), *local* variables, and *method* parameters.

The output from the program 1:

```
Class variable is 1
Instance variable is 2
Method parameter is 3
Local variable is 4
```

Whenever an instance method is invoked on an object, a hidden reference named **this** is always passed to the method. The **this** reference always refers to the object on which the method was invoked. This makes it possible for the code in the method to refer back to the object on which the method was invoked. The reference named **this** can be used to access the member variables hidden by the local variables or parameters having of the same name.

The program 2 illustrates the use of the **this** reference to access a hidden instance variable named `myVar` and a hidden class variable named `yourVar`.

The output from the program 2:

```
myVar parameter = 1
local yourVar variable = 2
Instance variable myVar = 3
Class variable yourVar = 4
```

[Program 1]

```
class JavaProg1 { // define the controlling class
    // declare and initialize variables
    A classVariable = 1;
    int instanceVariable = 2;

    public static void main(String[] args) { // main method
        System.out.println("Class variable is " + classVariable);

        // declare and initialize variable
        int localVariable = 4;

        // instantiate an object of the class to allow
```

```

// for access to instance variable and method
JavaProg1 obj = new JavaProg1();

System.out.println("Instance variable is "
                  + B );
obj.myMethod(3); // invoke the method
System.out.println("Local variable is " + localVariable);
} //end main

void myMethod(int methodParameter) {
    System.out.println("Method parameter is "
                      + methodParameter);
} // end myMethod
} // end JavaProg1 class.

```

[Program 2]

```

class JavaProg2 {
    int myVar = 0;
    static int yourVar = 0;

    // constructor with parameters named myVar and yourVar
    public JavaProg2(int myVar, int yourVar) {
        this.myVar = myVar;
        this.yourVar = yourVar;
    } // end constructor

    // method with parameter named myVar
    // and local variable named yourVar
    void myMethod(int myVar) {
        int yourVar = 2;
        System.out.println("myVar parameter = " + myVar);
        System.out.println("local yourVar variable = "
                          + yourVar);
        System.out.println("Instance variable myVar = "
                          + this.myVar);
        System.out.println("Class variable yourVar = "
                          + C );
    } // end myMethod

    public static void main(String[] args) {
        JavaProg2 obj = new JavaProg2(3, 4);
        obj.myMethod(1);
    } // end main method
} // end JavaProg2 class definition.

```

[Program 3]

```
public class JavaProg3 {
    public static void main (String args[]) {
        new Worker().doIt();
    } // end main()
} // end class JavaProg3

class Worker {
    void doIt() {
        Base myVar = new A();
        myVar.test();
        System.out.println("");
    } // end doIt()
} // end class Worker

class Base {
    public void test() {
        System.out.print("Base ");
    } // end class Base

class A extends Base {
    public void test() {
        System.out.print("A ");
    } // end test()
} // end class A
```

Subquestion 1

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

in the above program.

Answer group for A

- | | |
|---------------|-----------------|
| a) final int | b) int |
| c) static int | d) volatile int |

Answer group for B

- a) instanceVariable
- b) obj.instanceVariable
- c) obj.myMethod(instanceVariable)
- d) this.instanceVariable

Answer group for C

- | | |
|-----------------|---------------|
| a) myVar | b) this.myVar |
| c) this.yourVar | d) yourVar |

Subquestion 2

From the answer group below, select the correct output from the [Program 3].

Answer group

- | | |
|------------|--------------|
| a) "A " | b) "A Base " |
| c) "Base " | d) "Base A " |

Concerning questions **Q8** and **Q9**, **select one** of the two questions. Then, mark (S) in the selection area on the answer sheet, and answer the question.

If **two questions** are selected, **only the first question** will be graded.

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

[Program Description]

This is the program for playing the game called “paper, rock, scissors”. In this game, each child uses his/her hand to represent one of the three objects. A flat hand held in a horizontal position represents “paper”, a clenched fist represents “rock”, and index and middle fingers extended and separated represent “scissors”. The children face each other and display their choices. If the choices are the same, then the game is a tie. Otherwise, a win is determined by the following rules:

- Paper covers the rock; paper wins against rock.



- Rock breaks the scissors; rock wins against scissors.



- Scissors cut the paper; scissors win against paper.



The program uses `enum` to declare enumeration types. It provides a means of naming a finite set, and of declaring identifiers as elements of the set.

The following list shows a part of the variables used in the program.

`enum p_r_s` represents the player's choices

- `paper` – The player displays paper.
- `rock` – The player displays rock.
- `scissors` – The player displays scissors.
- `game` – The player outputs game status.
- `help` – The player outputs game's help.
- `quit` – The player quits the game.

`enum outcome` represents the game status (win, lose, tie, or error)

`player_choice` player's choice (see `enum p_r_s`)

`machine_choice` program generated random choice (paper, rock, scissors)

win_cnt	player's winning count
lose_cnt	player's losing count
tie_cnt	player's tied count

[Program]

```

/* The game of paper, rock and scissors. */
#include <ctype.h>    /* for isspace() */
#include <stdio.h>    /* for printf(), etc */
#include <stdlib.h>   /* for rand() and srand() */
#include <time.h>     /* for time() */

enum p_r_s {paper, rock, scissors, game, help, quit};
enum outcome {win, lose, tie, error};

typedef enum p_r_s p_r_s;
typedef enum outcome outcome;

outcome compare(p_r_s player_choice, p_r_s machine_choice);
void prn_final_status(int win_cnt, int lose_cnt);
void prn_game_status(int win_cnt, int lose_cnt, int tie_cnt);
void prn_help(void);
void report_and_tabulate(outcome result, int *win_cnt_ptr,
                        int *lose_cnt_ptr, int *tie_cnt_ptr);
p_r_s selection_by_machine(void);
p_r_s selection_by_player(void);

void prn_final_status(int win_cnt, int lose_cnt)
{
    if ( A )
        printf("CONGRATULATIONS - You won!\n\n");
    else if ( B )
        printf("A DRAW - You tied!\n\n");
    else
        printf("SORRY - You lost!\n\n");
}

void prn_game_status(int win_cnt, int lose_cnt, int tie_cnt)
{
    printf("\n%s\n%s%4d\n%s%4d\n%s%4d\n%s%4d\n\n",
        "GAME_STATUS:",
        " Win:      ", win_cnt,
        " Lose:     ", lose_cnt,
        " Tie:       ", tie_cnt,
        " Total:    ", win_cnt + lose_cnt + tie_cnt);
}

```



```

}

void prn_help(void)
{
    printf("\n%s\n",
        "The following characters can be used for input:\n"
        "  p   if button p is pressed, display paper\n"
        "  r   if button r is pressed, display rock\n"
        "  s   if button s is pressed, display scissors\n"
        "  g   if button g is pressed, print the game status (count of
win, lost, tie and total played)\n"
        "  h   if button h is pressed, print help, print this list\n"
        "  q   if button q is pressed, quit this game\n");
}

p_r_s selection_by_machine(void)
{
    return ((p_r_s) (rand() % 3));
}

p_r_s selection_by_player(void)
{
    int c;
    char str[256];
    p_r_s player_choice;

    player_choice = help;
    printf("Input p, r, or s:  ");
    fgets(str, 256, stdin);
    for (c=0; !isspace(str[c]) && c < strlen(str); c++) {
        switch (str[c]) {
            case 'p': player_choice = paper;
                       break;
            case 'r': player_choice = rock;
                       break;
            case 's': player_choice = scissors;
                       break;
            case 'g': player_choice = game;
                       break;
            case 'q': player_choice = quit;
                       break;
            default: break;
        }
    }
    return player_choice;
}

```

```

outcome compare(p_r_s player_choice, p_r_s machine_choice)
{
    outcome result;

    if ( C )
        return tie;
    switch (player_choice) {
        case paper:
            result = (machine_choice == rock) ? win : lose;
            break;
        case rock:
            result = (machine_choice == scissors) ? win : lose;
            break;
        case scissors:
            result = (machine_choice == paper) ? win : lose;
            break;
        default:
            printf("\nPROGRAMMER ERROR: Unexpected choice!\n\n");
            exit(1);
    }
    return result;
}

void report_and_tabulate(outcome result, int *win_cnt_ptr,
                        int *lose_cnt_ptr, int *tie_cnt_ptr)
{
    switch (result) {
        case win:
            D ;
            printf("%27sYou win.\n", "");
            break;
        case lose:
            ++*lose_cnt_ptr;
            printf("%27sYou lose.\n", "");
            break;
        case tie:
            ++*tie_cnt_ptr;
            printf("%27sA tie.\n", "");
            break;
        default:
            printf("\nPROGRAMMER ERROR: Unexpected result!\n\n");
            exit(1);
    }
}

```

```

int main(void)
{
    int      win_cnt = 0, lose_cnt = 0, tie_cnt = 0;
    outcome  result;
    p_r_s    player_choice, machine_choice;

    srand(time(NULL)); /* seed the random number generator */
    while ((player_choice = selection_by_player()) != quit)
    {
        switch (player_choice) {
            case paper:
            case rock:
            case scissors:
                machine_choice = selection_by_machine();
                result = compare(player_choice, machine_choice);
                report_and_tabulate(result, &win_cnt, &lose_cnt, &tie_cnt);
                break;
            case game:
                 ;
                break;
            case help:
                prn_help();
                break;
            default:
                printf("\nPROGRAMMER ERROR:  Cannot get to here!\n\n");
                exit(1);
        }
    }
    prn_game_status(win_cnt, lose_cnt, tie_cnt);
    prn_final_status(win_cnt, lose_cnt);
    return 0;
}

```

Subquestion

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

in the above program.

Answer group for A and B

- a) win_cnt < lose_cnt
- b) win_cnt <= lose_cnt
- c) win_cnt == lose_cnt
- d) win_cnt > lose_cnt

Answer group for C

- a) `player_choice < machine_choice`
- b) `player_choice <= machine_choice`
- c) `player_choice == machine_choice`
- d) `player_choice > machine_choice`
- e) `player_choice >= machine_choice`

Answer group for D

- a) `++*win_cnt_ptr`
- b) `++win_cnt_ptr`
- c) `--*win_cnt_ptr`
- d) `--win_cnt_ptr`
- e) `*win_cnt_ptr++`
- f) `win_cnt_ptr++`
- g) `*win_cnt_ptr--`
- h) `win_cnt_ptr--`

Answer group for E

- a) `prn_final_status(win_cnt, lose_cnt)`
- b) `prn_game_status(win_cnt, lose_cnt, tie_cnt)`
- c) `prn_help()`
- d) `report_and_tabulate(result, win_cnt, lose_cnt, tie_cnt)`

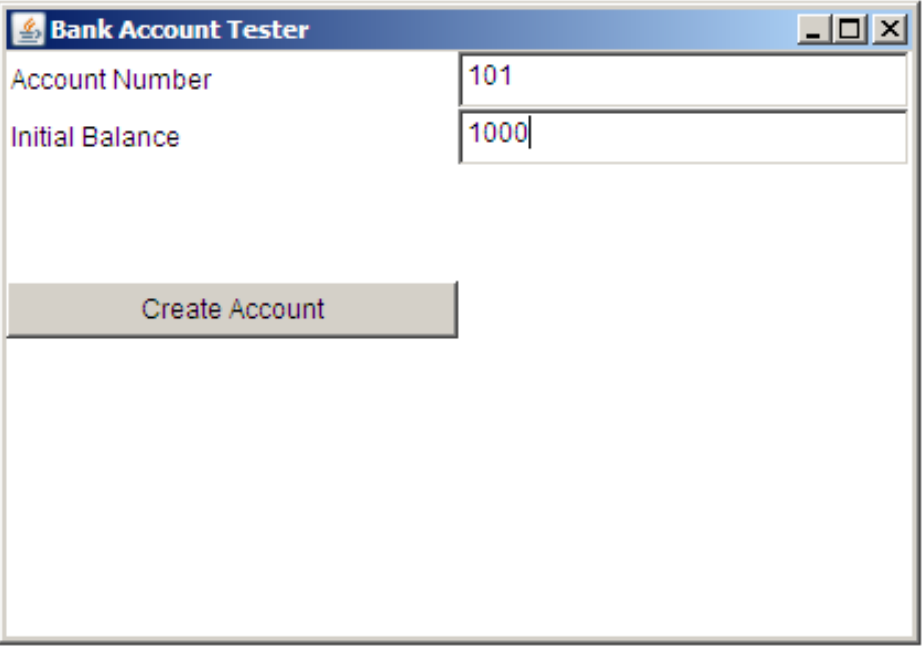
Q9 Read the following description of a Java program and the program itself, and then answer Subquestion.

A simple bank account simulation program is written in GUI java. This simulation allows the user to create a new account and test it. There are two methods given for account management: deposit and withdraw.

method	description
deposit	To add the amount to the bank account
withdraw	To subtract the amount from the bank account

There are two screens given: Create Account and Test Account. Create Account screen is used to create a new account number. Test Account screen is used to test deposit and withdrawal operations for the bank account.

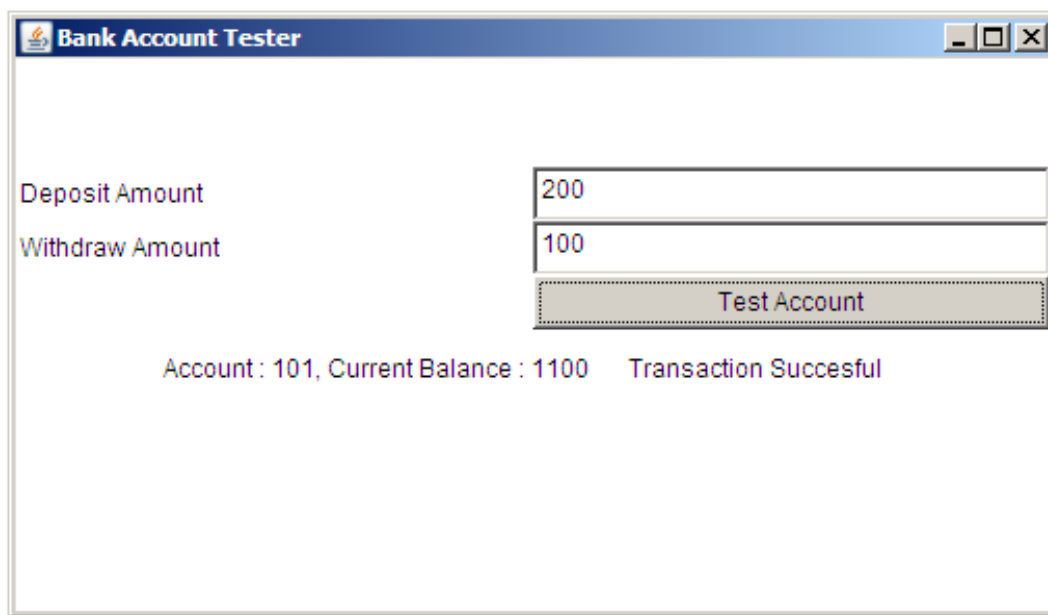
Diagram 1: Create Account



The screenshot shows a Java Swing window titled "Bank Account Tester". The window contains two text input fields. The first field is labeled "Account Number" and contains the text "101". The second field is labeled "Initial Balance" and contains the text "1000". Below these fields is a button labeled "Create Account". The window has a standard Mac OS X-style title bar with minimize, maximize, and close buttons.

Diagram 1 shows a sample input screen. The user is asked to enter a numeric account number and the initial balance. The user then clicks on the Create Account button.

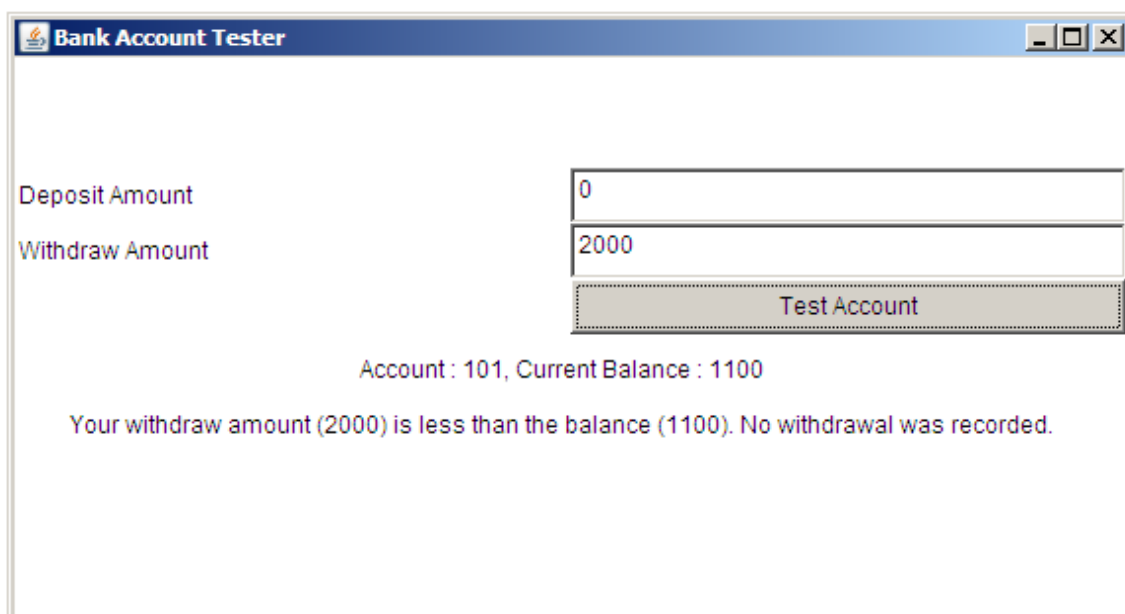
Diagram 2: Test Account – Transaction Successful



The screenshot shows a window titled "Bank Account Tester". It contains two input fields: "Deposit Amount" with the value "200" and "Withdraw Amount" with the value "100". Below these fields is a button labeled "Test Account". At the bottom of the window, a message reads: "Account : 101, Current Balance : 1100 Transaction Succesful".

In diagram 2, the user is asked to test the bank account by keying in the amount for deposit and withdrawal. If deposit only, enter “0” for withdrawal. If withdrawal only, enter “0” for deposit. After clicking the Test Account button, the user sees the message.

Diagram 3: Test Account – No withdrawal



The screenshot shows the same "Bank Account Tester" window. The "Deposit Amount" field is "0" and the "Withdraw Amount" field is "2000". The "Test Account" button is visible. Below the button, the message reads: "Account : 101, Current Balance : 1100". Below that, a second message states: "Your withdraw amount (2000) is less than the balance (1100). No withdrawal was recorded."

In diagram 3, the user has keyed in the amount for withdrawal larger than the balance available. The message shows that the withdrawal is not successful.

[Program]

```
//Program to create Graphical User Interface (GUI) for Bank Account
Simulation
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class GuiAccTest extends Frame implements ActionListener {
    Label lab=new Label("                                ");
    Label lab1=new Label("                                ");
    TextField t[]=new TextField [4];
    Label lbl[]=new Label [4];
    Button but=new Button("Create Account");
    Button but1=new Button("Test Account");
    BankAccount b;
    GuiAccTest(){
        addWindowListener(new NewWindowAdapter());
        setLayout(new GridLayout(2,0));
        Panel p=new Panel();
        Panel p1=new Panel();
        but.addActionListener(this);
        but1.addActionListener(this);
        p.setLayout(new GridLayout(5,2));
        p1.add(lab1);
        p1.add(lab);
        lbl[0]=new Label("Account Number");
        lbl[1]=new Label("Initial Balance");
        lbl[2]=new Label("Deposit Amount");
        lbl[3]=new Label("Withdraw Amount");
        for(int i=0;i<4;i++){
            t[i]=new TextField(10);
            p.add(lbl[i]);
            p.add(t[i]);
        }
        p.add(but);
        p.add(but1);
        but1.setVisible(false);
        lbl[2].setVisible(false);
        lbl[3].setVisible(false);
        t[2].setVisible(false);
        t[3].setVisible(false);
        add(p);
        add(p1);
    }
    String testAccount(int d_amt,int w_amt){
        String msg;
```

```

        A ;
        msg="Transaction Succesful";
        try {
            b.withdraw(w_amt);
        } catch(FundsInsufficientException fe){
            fe= B ;
            msg=String.valueOf(fe);
        }
        return msg;
    }
    C {
        String str=ae.getActionCommand();
        if(str.equals("Create Account")){
            b=new BankAccount(Integer.parseInt(t[0].getText()),
                               Integer.parseInt(t[1].getText()));
            but1.setVisible(true);
            lbl[2].setVisible(true);
            lbl[3].setVisible(true);
            t[2].setVisible(true);
            t[3].setVisible(true);
            but.setVisible(false);
            lbl[0].setVisible(false);
            lbl[1].setVisible(false);
            t[0].setVisible(false);
            t[1].setVisible(false);
            lab1.setText("Account:"+b.accnum+", Current Balance:"+b.amount);
            return;
        }
        else{
            lab.setText(testAccount(Integer.parseInt(t[2].getText()),
                                     Integer.parseInt(t[3].getText())));
            lab1.setText("Account:"+b.accnum+", Current
Balance:"+b.amount);
        }
    }
    public static void main(String arg[]){
        GuiAccTest at=new GuiAccTest();
        at.setTitle("Bank Account Tester");
        at.setSize(600,200);
        at.setVisible(true);
    }
}

class NewWindowAdapter extends WindowAdapter{
    public void windowClosing(WindowEvent we){
        System.exit(0);
    }
}

```



```

    }
}
class BankAccount{
    int accnum;
    int amount;
     {
        accnum=num;
        amount=amt;
    }
    public void deposit(int amt){
        amount=amount+amt;
    }
    public void withdraw(int amt) throws  {
        if(amt>amount)
            throw  ;
        else
            amount=amount-amt;
    }
}
class FundsInsufficientException extends  {
    int balance;
    int withdraw_amount;
    FundsInsufficientException(int bal,int w_amt){
        balance=bal;
        withdraw_amount=w_amt;
    }
     {
        return "Your withdraw amount (" + withdraw_amount
            + ")is larger than the balance (" + balance
            + "). No withdrawal was recorded.";
    }
}

```

Subquestion

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

Answer group for A

- a) amount += amt
- b) amount -= amt
- c) b.amount++
- d) b.deposit(d_amt)

e) `b.withdraw(d_amt)`

Answer group for B, E and F

- a) `FundsInsufficientException`
- b) `FundsInsufficientException()`
- c) `new FundsInsufficientException(amount, amt)`
- d) `new FundsInsufficientException(b.amount, w_amt)`
- e) `new FundsInsufficientException(d_amt, w_amt)`

Answer group for C

- a) `public void actionPerformed(ActionEvent ae)`
- b) `public void actionPerformed()`
- c) `public void actionPerformed(ActionEvent ae)`
- d) `public void actionPerformed()`
- e) `public void actionPerformed(ActionEvent ae)`

Answer group for D

- a) `BankAccount()`
- b) `BankAccount(int amt)`
- c) `BankAccount(int num)`
- d) `BankAccount(int num, int amt)`
- e) `BankAccount(int num, int amt, int deposit)`

Answer group for G

- a) `ActionEvent`
- b) `ActionListener`
- c) `ActionPerformed`
- d) `BankAccount`
- e) `Exception`

Answer group for H

- a) `public String display(String msg)`
- b) `public String toString()`
- c) `public String toString(String msg)`
- d) `public void display()`
- e) `public void toString()`