ESC101 : Fundamental of Computing

End Semester Exam **19 November 2008**

Name : **Roll No. :**

Section :

Note : Read the instructions carefully

1. You will lose 3 marks if you forget to write your name, roll number, and section.
2. There are **EIGHT** questions in this paper distributed among total five sheets of the answer sheet.
3. You have to write in the space provided. Please avoid over writing.
4. You will be given rough sheets.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Marks								

1. (marks = 5)
What is the value of `d` at the end of each of the following code fragments ? If you feel that there will be any compilation error, then mention that the cause of compilation error.

(a)
```
double d;

d = 2+11/9*4.5;
```

(b)
```
int i=10; byte b=12;

float d = b+i+1.3;
```

(c)
```
long d;

d = 1234567898765;
```

(d)
```
double A=11; byte b=2;

float d = (float)(A/b/2*4);
```

(e)
```
int d;

d = (int)3.4/1.1;
```

Answer :

(a)

(b)

(c)

(d)

(e)

2. (marks = 8)
Given two arrays `A` and `B` of size m and n respectively and $m \geq n$. The arrays are storing characters. We say that B appears in A if there exists a non-negative integer i such that for all j with $0 \leq j < n$, $A[i + j] = B[j]$. For example if `A=`$\{s, t, a, x, n, b, y\}$, and `B=`$\{a, x, n\}$, then `B` appears in `A`. Write a method `Appears` which takes two character arrays `A` and `B` as parameters and returns true if `B` appears in `A` and false otherwise. You may assume that `A.length` $\geq$ `B.length` is ensured by the method which calls the method `Appears`.
Answer :

3. (marks = 8)
The following method computes product of two matrices M and N and returns the reference to the final product matrix. Fill in the blanks appropriately.

```
// this method assumes that M and N are matrices
// and that columns of M = rows of N.
// It returns reference to the matrix which corresponds to M*N

public static ______________ multiply(int[ ][ ]  M, int[ ][ ]  N)
{

    int rows = _____________________;


    int columns = ___________________;


    int[][] R = new int[rows][columns];


    for(int r=0; _______________; r = r+1)
    {

         for(int c=0; _______________; c=c+1)

         {

              int product = _____________;


              for(int i=0;___________________;i=i+1)

              {

                  product =  __________ + ____________________________;

              }

              R[r][c]=  _______________;
         }
    }
    return R;
}
```

4. (marks=6)
Recall the class *Triangle* which was described in details during some lecture. For your assistance, the following was the skeleton of the *Triangle* class.

```
class Triangle
{    private Point P;
     private Point Q;
     Private Point R;

     public Triangle()
     {         .....        }

     public Triangle(Point A, Point B, Point C)
     {         ......       }

     public double Area()
     {         ......       }

     public double Perimeter()
     {         ......       }

}
```

We want to add one more non-static method `Encloses()` to this class. This method takes a point as parameter and returns true if the point is enclosed by the triangle and false otherwise. Note that a point lying on one of the sides of the triangle is also said to be enclosed by the triangle. Please fill in the blanks of the following skeleton of the method. **You may write at most one statement/expression in a blank.** You can't use && or || in the expression of `if` statement.

```
public boolean Encloses(Point O)
{

     ____________________________________________________________________________;


     ____________________________________________________________________________;


     ____________________________________________________________________________;


     if(_____________________________________________________________________________)


          return true;


     else return false;
}
```

5. (Marks = 3,5)
Draw the recursion tree for the following method when invoked with arguments $n = 10$, and $S =$ "". Also print the output of the method.

```
public static f(int n, String S)
{
        if(n<=2) System.out.println(S);
        else
        {        f(n-3, S+"-"+n);

                 if(n>=4) f(n/2,S+"-"+(n/2));
        }
}
```

6. (7 marks)
There is a function G defined for positive integers as follows.

$$G(n) = \begin{cases} 4 & \text{if} \quad 1 \leq n \leq 4 \\ G(n-1) + G(n-2) + 2G(n-3) \text{ - } 2 & \text{if} \quad n > 4 \end{cases}$$

Write a method for computing $G(n)$ which performs at most $c\ n$ instructions for some constant c.

```
public static  int G(int n)
{




}
```

7. (Marks = 1,3,3,1)

Given a grid with height n and width $2n$, We want to enumerate all those shortest paths from origin $(0, 0)$ to $(2n, n)$ which satisfy the constraint :

Each point of the path is either on or below the diagonal.

See figure given below for better understanding. The program enumerates a path so that it prints `-up` for a step in upward direction and prints `-right` for a step in right direction. For example for $n = 2$, the output should be

```
-right-right-up-right-right-up
-right-right-right-up-right-up
-right-right-right-right-up-up
```

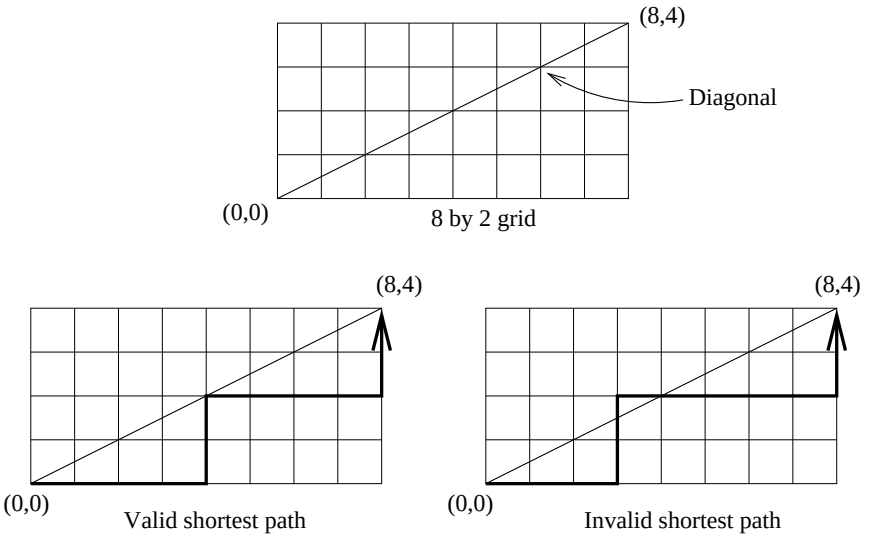

Figure 1: The grid, an example of valid path and an example of invalid path

Please fill in the blanks of the following program carefully. The program reads the value of n from command line.

```
class special_paths
{
   public static void PathS(int r, int u, String S)
   {
       if(_____________________) System.out.println(S);

       else
       {
             ____________________________________________________________;


             ____________________________________________________________;

       }
   }

   public static void main(String args[])
   {   int n = Integer.parseInt(args[0]);
       if(n<1) System.out.println("Invalid input");
       else
            PathS(___________________________);
   }
}
```

8. (10) The idea underlying binary search is a very useful idea and can be used for a variety of problems. We showed that we can use it for the following problem. Given an array A which stores 0's and 1's such that all 0's precede all 1's, find the smallest index of the array which stores 1. We are going to use similar idea for the following problem.

   There is an array A storing n distinct integers. The numbers are arranged such that there is a non-negative integer $p< n$ such that $A[0], ...A[p]$ is a strictly increasing sequence and $A[p], ..., A[n-1]$ is a strictly decreasing sequence of numbers. The following code prints the value of p. Fill in the blanks accordingly.

```
public static void Largest_value(int[] A)
{   int n = A.length;
    int mid = 0;

    if(_____________) System.out.println(``The value of p = 0'');

    else
    {    if(_____________) System.out.println(``The value of p =''+__________);

         else
         {     boolean p_is_found = false;

               int left = ___________________;

               int right = n-2;

               while(_________________________)
               {

                    mid = (left+right)/2;

                    if(A[mid]>_____________)
                    {

                         if(___________________)     ____________________________;


                         else   ____________________________________________
                    }

                    else  ___________________________;
               }

               System.out.println("The value of p is "+mid);

         }
    }
}
```

9.