Computer Science Project

(2017-18)

Name: Apurva Nagar

Class: XII - G

Internal Examiner's sign: _____________

External Examiner's sign: _____________

Acknowledgement

I would like to express my special thanks of gratitude to my teacher Mr. Akash Gupta as well as our principal Mrs. Jyoti Kashyap who gave me the golden opportunity to do this wonderful project on the topic Java Programs, which also helped me in doing a lot of research and I came to know about so many new things. I am really thankful to them.

Secondly I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

Index

Question 1

A Circular Prime is a prime number that remains prime under cyclic shifts of its digits. When the leftmost digit is removed and replaced at the end of the remaining string of digits, the generated number is still prime. The process is repeated until the original number is reached again.
A number is said to be prime if it has only two factors 1 and itself.

Example:

131
311
113

Hence, 131 is a circular prime.
Test your program with the sample data and some random data:

Example 1

INPUT : *N = 197*
OUTPUT:

```
197
971
719
197 IS A CIRCULAR PRIME
```

Example 2

INPUT : *N = 1193*
OUTPUT:

```
1193
1931
9311
3119
1193 IS A CIRCULAR PRIME
```

Example 3

INPUT : *N = 29*
OUTPUT:

```
29
92
29 IS NOT A CIRCULAR PRIME
```

Answer 1

```
public class CircularPrime
{
    public static void main(int num)
    {
        int temp = num;
        while(isPrime(temp) && ((temp =
circularLeftRotation(temp)) != num))
        {
            System.out.println(temp);
        }
        if(num == temp && isPrime(num))
                System.out.println("Circular Prime " + temp);
        else
                System.out.println("Not Circular Prime");
    }
    public static int  circularLeftRotation(int num)
    {
        int unitDigit = num%10;
        num = num/10;
        int temp = 1;
        while(num/temp != 0)
            temp*=10;
        return num+unitDigit*temp;
    }
    public static boolean isPrime(int num)
    {
        int i;
        int loopCount = (int)Math.sqrt(num);
        for(i=2;i<=loopCount && (num%i !=0);i++ );
        if(i==loopCount+1)
            return true;
        else
            return false;
    }
}
```

Algorithm:

1. Perform circular shift on the number one digit at a time
    a. Extract last digit
    b. Count number of remaining digits
    c. Add remaining number to last digit times ten to the power digits
2. For each circular shift, check if it is a prime or not.
3. After all circular shifts are done, print a appropriate message.

Output:

```
INPUT: N = 29
OUTPUT:
29
92
29 IS NOT A CIRCULAR PRIME.
```

Question 2

Write a program to declare a square matrix A[][] of order (M x M) where 'M' must be greater than 3 and less than 10. Allow the user to input positive integers into this matrix. Perform the following tasks on the matrix:

1. *kSort the non-boundary elements in ascending order using any standard sorting technique and rearrange them in the matrix.*
2. *Calculate the sum of both the diagonals.*
3. *Display the original matrix, rearranged matrix and only the diagonal elements of the rearranged matrix.*

Test your program with the sample data and some random data:

Example 1

```
INPUT :M = 4
9   2   1   5
8   13  8   4
15  6   3   11
7   12  23  8
OUTPUT:
ORIGINAL MATRIX
9   2   1   5
8   13  8   4
15  6   3   11
7   12  23  8
REARRANGED MATRIX
9   2   1   5
8   3   6   4
15  8   13  11
7   12  23  8
DIAGONAL ELEMENTS
9           5
    3   6
    8   13
7           8
```

Answer 2

```
import java.util.*;
class SqrMat
{
    static void main(int M)
    {
        Scanner sc = new Scanner(System.in);

        if(!(M>3&&M<10)) {return;}
        int[][] mat = new int[M][M];
        int[][] newMat = new int[M][M];

        for(int i =0; i<=M-1; i++)
            for(int j =0; j<=M-1; j++)
            {
                newMat[i][j] = mat[i][j] = sc.nextInt();
            }

        System.out.println("ORIGINAL MATRIX");
        for(int i =0; i<=M-1; i++)
        {
            for(int j =0; j<=M-1; j++)
            {
                System.out.print(mat[i][j] + " ");
            }
            System.out.println();
        }

        Rearrange(newMat);

        System.out.println("REARRANGED MATRIX");
        for(int i =0; i<=M-1; i++)
        {
            for(int j =0; j<=M-1; j++)
            {
                System.out.print(newMat[i][j] + " ");

            }
            System.out.println();
        }

        System.out.println("DIAGONAL ELEMENTS");

        int s =0;

        for(int i =0; i<=M-1; i++){
            for(int j =0; j<=M-1; j++)
            {
                if(i+j == M-1 || i==j)
                    System.out.print(newMat[i][j] + " ");
                else  System.out.print(" ");
                s+= newMat[i][j];
            }
            System.out.println();
        }
        System.out.println("SUM OF DIAGONAL ELEMENTS : " + s);
    }

    static void Rearrange(int[][] mat)
    {
        int c=0;
        int M = mat.length;
        int[] arr  = new int[(M-2)*(M-2)];

        for(int i =1; i<=M-2; i++){
            for(int j =1; j<=M-2; j++)
```

```
                {
                    arr[c++] = mat[i][j];
                }
             }

            //Bubble sort

            for(int x = 0; x < arr.length; x++)
            {
                for(int y = 0; y < arr.length -1 - x; y++)
                {
                    if(arr[y] > arr[y+1])

                    {
                        arr[y] += arr[y+1];
                        arr[y+1] = arr[y] - arr[y+1];
                        arr[y] -= arr[y+1];
                    }
                }
            }

            c = 0;
            for(int i =1; i<=M-2; i++){
                for(int j =1; j<=M-2; j++)
                {
                    mat[i][j] = arr[c++];
                }
            }

        }

    }
```

Algorithm:

static void main(int M)

1) START
2) Take integer input and store as M.
3) Check if M>3 and M<10 is true, if not,display INVALID INPUT and END.
4) Declare 2 2D matrices mat and newMat.
5) Fill them both with the same user input data.
6) display mat with heading ORIGINAL MATRIX by looping through it and adding line exits for new row.
7) Call Rearrange() with newMat as input.
8) display newMat with heading REARRANGED MATRIX by looping through it and adding line exits for new row.
9) display heading DIAGONAL ELEMENTS.
10) Declare an integer s to store sum. Initialise as 0.
11) Start a nested loop through newMat with i and j (o to M-1) tracking row and column indices, while adding line exits for each new row.
12) If i+j equals M-1 OR i equals j, display the value stored at that index, else display blank.
13) Add the same value to s if condition of 11.1. Is true.
14) display sum of diagonal elements.
15) END

static void Rearrange(int[][] mat)

1) START
2) Declare int c, M. Initialise with 0 and inputted matrix length respectively.
3) Declare an array arr of M-2 by M-2.
4) Store values of matrix linearly in arr.
5) Sort values in arr.
6) Replace values of inputted matrix with sorted values in the same order as extracted.
7) END OF ALGORITHM FOR Rearrange().

Output:

```
INPUT: M = 4
9   2   1   5
8   13  8   4
15  6   3   11
7   12  23  8
OUTPUT:
ORIGINAL MATRIX
```

```
9   2   1   5
8   13  8   4
15  6   3   11
7   12  23  8
REARRANGED MATRIX
9   2   1   5
8   3   6   4
15  8   13  11
7   12  23  8
DIAGONAL ELEMENTS
9           5
    3   6
    8   13
7           8
SUM OF THE DIAGONAL ELEMENTS = 59
```

Question 3 -

Write a program to accept a sentence which may be terminated by either '.', '?' or '!' only. The words may be separated by more than one blank space and are in UPPER CASE. Perform the following tasks:

1. *Find the number of words beginning and ending with a vowel.*
2. *Place the words which begin and end with a vowel at the beginning, followed by the remaining words as they occur in the sentence.*

Test your program with the sample data and some random data:

Example 1

```
INPUT: ANAMIKA AND SUSAN ARE NEVER GOING TO QUARREL ANYMORE.
OUTPUT: NUMBER OF WORDS BEGINNING AND ENDING WITH A VOWEL = 3
ANAMIKA ARE ANYMORE AND SUSAN NEVER GOING TO QUARREL
```

Example 2

```
INPUT: HOW ARE YOU@
OUTPUT: INVALID INPUT
```

Answer 3:

```
import java.util.*;
class Q3
{
    static void main(String s)
    {
        String[] arr;
        String ns = "";
        StringTokenizer st = new StringTokenizer(s, ".,?! ");
        int c = st.countTokens();

        if(".,?! ".indexOf(s.charAt(s.length()-1)) < 0){

            System.out.println("INVALID INPUT");
            return;
        }

    arr = new String[c];
    int i;
    for(i = 0; i < c; i++)
    {
        arr[i] = st.nextToken();
    }

    int cv = 0; //counts vowels
    for(i = 0; i < c; i++)
    {
        if("AIUEOaiueo".indexOf(arr[i].charAt(0)) > -1 &&
        "AIUEOaiueo".indexOf(arr[i].charAt(arr[i].length()-1)) >
-1)
        {
            ns = ns + " " +  arr[i]; cv++;
        }
    }
    for(i = 0; i < c; i++)
    {
        if(!("AIUEOaiueo".indexOf(arr[i].charAt(0)) > -1))
            ns = ns + " " + arr[i];
    }
    System.out.println("NUMBER OF WORDS BEGINNING AND ENDING
WITH A VOWEL = " + cv);
     System.out.println(ns);

}
}
```

Algorithm:

```
static void main(String s)
    1) START
    2) Input string from user. Store in s.
    3) Declare array arr, int c and cv (as 0), and string ns (as blank).
    4) Check if last character is .,?! If not display INVALID INPUT and END
    5) Separate words and store them in array arr. Store its length in int c.
    6) Start loop through arr, till its length.
    7) If word at current iteration has first and last character as vowel, store it to the right of
       contents of string variable ns.
    8) If condition is 5.1 is true, also add 1 to cv.
    9) Start loop through arr, till its length.
    10) If word at current iteration does not have first and last character as vowel, store it to the
        right of contents of string variable ns.
    11) display ns.
    12) END
```

Output:

```
INPUT: ANAMIKA AND SUSAN ARE NEVER GOING TO QUARREL ANYMORE.
OUTPUT: NUMBER OF WORDS BEGINNING AND ENDING WITH A VOWEL = 3
ANAMIKA ARE ANYMORE AND SUSAN NEVER GOING TO QUARREL
```

Question 4

Given two positive numbers M and N, such that M is between 100 and 10000 and N is less than 100. Find the smallest integer that is greater than M and whose digits add up to N. For example, if M=100 and N=11, then the smallest integer greater than 100 whose digits add up to 11 is 119.
Write a program to accept the numbers M and N from the user and print the smallest required number whose sum of all its digits is equal to N. Also, print the total number of digits present in the required number. The program should check for the validity of the inputs and display an appropriate message for an invalid input.
Test your program with the sample data and some random data:

Example 1

INPUT:

```
M = 100
N = 11
```

OUTPUT:

```
The required number = 119
Total number of digits = 3
```

Example 2

INPUT:

```
M = 1500
N = 25
```

OUTPUT:

```
The required number = 1699
Total number of digits = 4
```

Example 3

INPUT:

```
M = 99
N = 11
```

OUTPUT:

```
INVALID INPUT
```

Example 4

INPUT:

```
M = 112
N = 130
```

OUTPUT:

```
INVALID INPUT
```

Answer 4:

```
import java.util.*;
class Q4
{
    static void main()
    {
        System.out.println("INPUT M AND N");
        Scanner sc = new Scanner(System.in);
        int M = sc.nextInt(); int N = sc.nextInt();

        if (!(M>= 100 && M <=10000 && N<100) && N>0)
        {
            System.out.println("INVALID INPUT");
            return;
        }

        int i = M+1;
        while(true)
        {
            if(DigSum(i++) == N) break;
        }
        System.out.println("The required number = " + (i-1));
        System.out.println("Total number of digits = " + (i-
1+"").length());
    }

    static int DigSum(int n)
    {
        int s = 0;
        for(int i = 0; i < (n+"").length(); i++)
        {
            s+= Integer.parseInt(""+(n+"").charAt(i));
        }
        return s;
    }
}
```

Algorithm:

```
static void main()
    1) START
    2) Input M and N.
    3) Check if M is within 100 and 100, and N is within 0 and 100. If not,display INVALID
       INPUT and END.
    4) Start loop with int i as M+1.
    5) Call DigSum() with i as input.
    6) If value returned in 3.1 is equal to N, exit out of loop, else add 1 to i.
    7) display i-1 as the required number.
    8) display number of digits of i-1.
    9) END

static int DigSum(int n)
    1) START
    2) Declare int s and initialise as 0.
    3) Coverrt inputted n to string.
    4) Loop through its characters.
    5) Convert character to integer.
    6) Add it to s.
    7) Return s.
    8) END OF ALGORITHM for DigSum(int n)
```

Output:

```
INPUT:
M = 1500
N = 25
OUTPUT:
The required number = 1699
Total number of digits = 4
```

Question 5

Write a program to declare a square matrix A[][] of order M×M where 'M' is the number of rows and the number of rows and the number of columns, such that M must be greater than 2 and less 10. Accept the value M as user input. Display an appropriate message for an invalid input. Allow the user to input integers into the matrix. Perform the following tasks:

1. *Display the original matrix.*
2. *Rotate the matrix 90° clockwise as shown below:*

Original matrix			*Rotated matrix*		
1	2	3	7	4	1
4	5	6	8	5	2
7	8	9	9	6	3

3. *Find the sum of the elements of the four corners of the matrix.*

Example 1

INPUT:

```
M = 3
3  4  9
2  5  8
1  6  7
```

OUTPUT:

```
ORIGINAL MATRIX
3  4  9
2  5  8
1  6  7
MATRIX AFTER ROTATION
1  2  3
6  5  4
7  8  9
Sum of the corner elements = 20
```

Example 2

INPUT:

```
M = 4
1  2  4  9
2  5  8  3
1  6  7  4
3  7  6  5
```

OUTPUT:

```
ORIGINAL MATRIX
1  2  4  9
2  5  8  3
1  6  7  4
3  7  6  5
MATRIX AFTER ROTATION
```

```
3  1  2  1
7  6  5  2
6  7  8  4
5  4  3  9
```

Sum of the corner elements = 18

Example 3

INPUT:

M = 14

OUTPUT:

SIZE OUT OF RANGE

Example 4

INPUT:

M = 112

N = 130

OUTPUT:

INVALID INPUT

Answer 5

```
import java.util.*;
class Rotate90
{
    static void main()
    {
        System.out.println("Enter Input");
        Scanner sc = new Scanner(System.in);
        int M = sc.nextInt();

        if (!(M>=3 && M <=9))
        {
            System.out.println("INVALID INPUT");
            return;
        }
        int[][] mat = new int[M][M];
        int[][] newmat = new int[M][M];
        int s = 0;

        for(int i =0; i<=M-1; i++)
        {
            for(int j =0; j<=M-1; j++)
            {
                newmat[j][(M-1) - i] = mat[i][j] = sc.nextInt();
//Applies transform during initialization
                if((i==0 &&(j==0||j==M-1)) || (i==M-1
&&(j==0||j==M-1)) ) s+=  mat[i][j]; //Corner elements
            }
        }
        System.out.println("ORIGINAL MATRIX");

        for(int i =0; i<=M-1; i++)
        {
            for(int j =0; j<=M-1; j++)
            {
                System.out.print(mat[i][j]);

            }
            System.out.println();
        }
        System.out.println("MATRIX AFTER ROTATION");

        for(int i =0; i<=M-1; i++){
            for(int j =0; j<=M-1; j++)
            {
                System.out.print(newmat[i][j]);
            }
            System.out.println();
        }
        System.out.println("Sum of the corner elements = " +s);
    }
}
```

Algorithm:

static void main()

1) START
2) Input M.
3) Check that M is within 3 and 9. If not, display INVALID INPUT and END.
4) Declare integer 2d matrices of M by M mat and newmat. Declare int s and initialise at 0.
5) Start nested loop in i,j with j 0 to M-1 for each iteration of i from 0 to M-1.
6) Take user input at (i,j)th index in mat.
7) Store it at (j, M-1-i)th index in newmat.
8) If i,j represent a corner element, add input to s.
9) display mat with heading ORIGINAL MATRIX.
10) display newmat with heading MATRIX AFTER ROTATION.
11) display s.
12) END

Output:

```
INPUT :
M = 4
1  2  4  9
2  5  8  3
1  6  7  4
3  7  6  5
OUTPUT :
ORIGINAL MATRIX
1  2  4  9
2  5  8  3
1  6  7  4
3  7  6  5
MATRIX AFTER ROTATION
3  1  2  1
7  6  5  2
6  7  8  4
5  4  3  9
Sum of the corner elements = 18
```

Question 6

Write a program to accept a sentence which may be terminated by either '.' Or '?' only. The words are to be separated by a single blank space. Print an error message if the input does not terminate with '.' Or '?'. You can assume that no word in the sentence exceeds 15 characters, so that you get a proper formatted output.

Perform the following tasks:

1. *Convert the first letter of each word to uppercase.*
2. *Find the number of vowels and consonants in each word and display them with proper headings along with the words.*

Test your program with the following inputs.

Example 1

INPUT: *Intelligence plus character is education.*

OUTPUT:

Intelligence Plus Character Is Education

WORD	VOWELS	CONSONANTS
Intelligence	5	7
Plus	1	3
Character	3	6
Is	1	1
Education	5	4

Example 2

INPUT: *God is great.*

OUTPUT:

God Is Great

WORDS	VOWELS	CONSONANTS
God	1	2
Is	1	1
Great	2	3

Example 3

INPUT: *All the best!*

OUTPUT:

Invalid Input.

Answer 6:

```
import java.util.*;
class Q3_ISC2017
{
    int countVowel(String s) // Function to count no. of vowels in a
word
    {
        s = s.toUpperCase();
        int count = 0;
        char ch;
        for(int i=0; i<s.length(); i++)
        {
            ch = s.charAt(i);
            if(ch=='A' || ch=='E' || ch=='I' || ch=='O' || ch=='U')
            {
                count++;
            }
        }
        return count;
    }

    String convert(String s) // Function to convert 1st character to
UpperCase
    {
        char ch = s.charAt(0); // Extracting the first character
        ch = Character.toUpperCase(ch); // Converting that character
to UpperCase
        String f = ch + s.substring(1); // Adding it with the rest of
the string
        return f;
    }

    String addSpace(String s, int max) // Function for addng extra
space to make every word equal in length
    {
        int x = max-s.length();
        for(int i=1; i<=x; i++)
        {
            s = s+" ";
        }
        return s;
    }
    public static void main(String args[])
    {
        Q3_ISC2017 ob = new Q3_ISC2017();
        Scanner sc=new Scanner(System.in);

        System.out.print("Enter a sentence : ");
        String s=sc.nextLine();
        int l = s.length();
        char last = s.charAt(l-1); // Extracting the last character

        /* Checking whether the sentence ends with '.' or '?' or not
*/
        if(last != '.' && last != '?')
        {
            System.out.println("Invalid Input. End a sentence with
either '.' or '?'");
        }
        else
        {
            StringTokenizer str = new StringTokenizer(s," .?");
            int x = str.countTokens();
            String ans="";
            String word[]=new String[x];
            int vow, con, max=0;

            for(int i=0; i<x; i++)
k            {
                word[i] = str.nextToken(); // Extracting words and
saving them in an array
                ans = ans + " " + ob.convert(word[i]);
                if(word[i].length()>max)
                {
                    max = word[i].length();
                }
            }
            System.out.println("Sentence = "+ans.trim());

            String y=ob.addSpace("Word",max);
            System.out.println(y+"\tVowels\tConsonant");
            for(int i=0; i<x; i++)
            {
                vow = ob.countVowel(word[i]);
                con = word[i].length()-vow; // Consonant = Length - 
Vowel
                y = ob.addSpace(word[i],max);
                System.out.println(y+"\t"+vow+"\t"+con);
            }
        }
    }
}
```

Algorithm:

```
static void main()
    1)  START
    2)  Input string from user. Store in s.
    3)  Declare array arr and cv, int c (as 0), and string ns (as blank).
    4)  Check if last character is .? If not display INVALID INPUT and END
    5)  Store number of words in int c.
    6)  Start loop through arr with i 0 to its length.
    7)  Capitalise first letter of each word.
    8)  Start loop through letters of the word.
    9)  If letter is a vowel, add 1 to value at i th index of cv.
    10) display headings WORDS VOWELS CONSONANTS.
    11) Start loop with i from 0 to c.
    12) display i th index values of arr, cv, and difference between them.
    13) Add line exit every iteration.
    14) END
```

Output:

```
INPUT: God is great.
OUTPUT:
God Is Great
WORD                    VOWELS                  CONSONANTS
God                     1                       2
Is                      1                       1
Great                   2                       3
```

Question 7

A composite Magic number is a positive integer which is composite as well as a magic number.

Composite number: *A composite number is a number which has more than two factors. For example: 10 Factors are: 1,2,5,10*

Magic number: *A Magic number is a number in which the eventual sum of the digit d is equal to 1. For example: 28 = 2+8=10= 1+0=1*

Accept two positive integers m and n, where m is less than n as user input. Display the number of composite magic integers that are in the range between m and n (both inclusive) and output them along with frequency, in the format specified below:

Example 1:

INPUT:

m = 10

n = 100

OUTPUT:

```
THE COMPOSITE MAGIC INTEGERS ARE:
10, 28, 46, 55, 64, 82, 91, 100
FREQUENCY OF COMPOSITE MAGIC INTEGERS IS: 8
```

Example 2:

INPUT:

m = 1200

n = 1300

OUTPUT:

```
THE COMPOSITE MAGIC INTEGERS ARE:
1207, 1216, 1225, 1234, 1243, 1252, 1261, 1270, 1288
FREQUENCY OF COMPOSITE MAGIC INTEGERS IS: 9
```

Example 3:

INPUT:

m = 120

n = 99

OUTPUT:

```
INVALID INPUT
```

Answer 7:

```
class CompMagic
{
      void main(int m, int n)
      {
            int cnt = 0;
            if (m<n)
            {
                  System.out.print("The composite magic numbers are: ");
                  for (int i=m; i<=n; i++)
                  {
                        if (isComposite(i) && isMagic(i))
                        {
                              cnt++;
                              System.out.print(i + ", ");
                        }
                  }
                  System.out.println("\n Frequency of composite magic
numbers: " + cnt);
            }
            else
            {
                  System.out.println("Invalid input");
                  System.exit(0);
            }
      }

      boolean isComposite(int x)
      {
            int c = -1;
            for (int i=2; i<x; i++)
            {
                  if(x % i == 0)
                        c++;
            }
            if(c == -1)
                  return false;
            else
                  return true;
      }
boolean isMagic(int x)
      {
            int sum = 0; int num = x;
            while(num > 9)
            {
                  sum = num;
                  int s = 0;
                  while(sum != 0)
                  {
                        s = s + (sum%10);
                        sum = sum / 10;
                  }
                  num = s;
            }
            if (num == 1)
                  return true;
            else
                  return false;
      }
}
```

Output:

```
INPUT:
m = 10
n = 100
OUTPUT:
THE COMPOSITE MAGIC INTEGERS ARE:
10, 28, 46, 55, 64, 82, 91, 100
FREQUENCY OF COMPOSITE MAGIC INTEGERS IS: 8
```

Question 8

Write a program to declare a square matrix A[][] of order MXM where M is an positive integer and represents row and column. M should be greater than 2 and less than 10.Accept the value of M from user. Display an appropriate message for invalid input.

Perform the following task:

1. *Display the original matrix*
2. *Check if the given matrix is symmetric or not. If the element of the ith row and jth column is same as element of the jth row and ith column.*
3. *Find the sum of the left and right diagonal of the matrix and display them*

Example 1

INPUT : *M = 3*

```
1    2    3
2    4    5
3    5    6
```

OUTPUT :

```
ORIGINAL MATRIX
1    2    3
2    4    5
3    5    6
THE GIVEN MATRIX IS SYMMETRIC
The sum of the left diagonal = 11
The sum of the right diagonal = 10
```

Example 2

INPUT : *M = 4*

```
7    8    9    2
4    5    6    3
8    5    3    1
7    6    4    2
```

OUTPUT :

```
ORIGINAL MATRIX
7    8    9    2
4    5    6    3
8    5    3    1
7    6    4    2
THE GIVEN MATRIX IS NOT SYMMETRIC
The sum of the left diagonal = 17
The sum of the right diagonal = 20
```

Example 3

INPUT : *M = 22*

OUTPUT : *THE MATRIX SIZE IS OUT OF RANGE*

Answer 8:

```
import java.io.*;
class SymetricMatrix
{
   public static void main(String args[])throws IOException
   {
       BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));

       System.out.print("Enter the number of elements : ");
       int m=Integer.parseInt(br.readLine());
       int A[][]=new int[m][m];

       if(m>2 && m<10) // Checking for valid input of rows and columns
size
       {
           System.out.println("\nInputting the elements in the Matrix:
\n");
           for(int i=0;i<m;i++)
           {
               for(int j=0;j<m;j++)
               {
                   System.out.print("Enter the elements : ");
                   A[i][j]=Integer.parseInt(br.readLine());
               }
           }

           /* Printing the Original Matrix */
           System.out.println("\nThe Original Matrix is : ");
           for(int i=0;i<m;i++)
           {
               for(int j=0;j<m;j++)
               {
                   System.out.print(A[i][j]+"\t");
               }
               System.out.println();
           }

           /* Checking whether the matrix is symmetric or not */
           int flag = 0;
           for(int i=0;i<m;i++)
           {
               for(int j=0;j<m;j++)
               {
                   if(A[i][j] != A[j][i])
                   {
                       flag = 1; // Setting flag = 1 when elements do
not match
                       break;
                    }
               }
           }

           if(flag == 1)
                System.out.println("\nThe given Matrix is Not
Symmetric");
           else
                System.out.println("\nThe given Matrix is Symmetric");

            /* Finding sum of the diagonals */
            int ld = 0, rd = 0;
            for(int i=0;i<m;i++)
            {
                for(int j=0;j<m;j++)
                {
                    if(i == j) // Condition for the left diagonal
                    {
                        ld = ld + A[i][j];
                    }
                    if((i+j) == (m-1)) // Condition for the right
diagonal
                    {
                        rd = rd + A[i][j];
                    }
                }
            }

            System.out.println("The sum of the left diagonal = "+ld);
            System.out.println("The sum of the right diagonal = "+rd);
       }

       else
            System.out.println("The Matrix Size is Out Of Range");
   }
}
```

Algorithm:

```
static void main()
    1) START
    2) Input M.
    3) Check that M is within 3 and 9. If not, display that size if out of range and END.
    4) Declare integer 2d matrix of M by M as mat.
    5) Declare int variables lds, rds with initial values as 0. And a boolean flag with true as
       default value.
    6) Take user input and store values in mat.
    7) display mat with heading ORIGINAL MATRIX.
    8) Loop through mat with variables i,j tracking row,column indices.
    9) If value at (i,j) is not equal to value at (j,i) set flag to false.
    10) If i+j equals M-1, add value at (i,j) to rds.
    11) If i equals j, add value at (i,j) to lds.
    12) If flag is true, display matrix is symmetric, else display that it is not symmetric.
    13) display lds.
    14) display rds.
    15) END
```

Output:

```
INPUT       :       M = 3
1     2     3
2     4     5
3     5     6
OUTPUT      :
ORIGINAL MATRIX
1     2     3
2     4     5
3     5     6
THE GIVEN MATRIX IS SYMMETRIC
The sum of the left diagonal = 11
The sum of the right diagonal = 10
```

Question 9

Write a program to accept a sentence which may be terminated by either '.', '?', or '!' only. Any other character may be ignored. The words may be separated by more than one blank space and are in UPPER CASE.

Perform the following tasks.

1. *Accept the sentence and reduce all the extra blank space between two words to a single blank space.*
2. *Accept a word from the user which is part of the sentence along with its position number and delete the word and display the sentence.*

Test your program with the sample data and some random data.

Example 1

INPUT: *A   MORNING WALK IS A IS BLESSING FOR   THE WHOLE DAY.*

WORD TO BE DELETED: IS

WORD POSITION IN THE SENTENCE: 6

OUTPUT: *A MORNING WALK IS A BLESSING FOR THE WHOLE DAY.*

Example 2

INPUT: *AS YOU   SOW, SO   SO YOU REAP.*

WORD TO BE DELETED: SO

WORD POSITION IN THE SENTENCE: 4

OUTPUT: *AS YOU SOW, SO YOU REAP.*

Example 3

INPUT: *STUDY WELL ##*

OUTPUT: *INVALID INPUT*

Answer 9:

```
import java.util.*;
class RemoveWord_ISC2017
{
    public static void main (String args[])
    {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a sentence : ");
        String s = sc.nextLine();
        s = s.toUpperCase();
        int l = s.length();
        char last = s.charAt(l-1); // Extracting the last character

        /* Checking whether the sentence ends with '.' or '?' or not
*/
        if(last != '.' && last != '?' && last != '!')
        {
            System.out.println("Invalid Input. End a sentence with
either '.', '?' or '!' only");
        }
        else
        {
            StringTokenizer str = new StringTokenizer(s," .?!");
            int c = str.countTokens();
            String w="",ans = "";
            System.out.print("Enter the word to delete : ");
            String del = sc.next();
            System.out.print("Enter the word position is the sentence
: ");
            int x = sc.nextInt();

            if(x<1 || x>c) // Checking whether integer inputted is
acceptable or not
            {
                System.out.println("Sorry! The word position entered
is out of range");
            }
            else
            {
                for(int i=1; i<=c; i++)
                {
                    w = str.nextToken();
                    /* Skipping if the word to delete and the position
matches */
                    if(w.equals(del)==true && i == x)
                        continue;
                    ans = ans + w + " ";
                }
                System.out.print("Output : "+ans.trim()+last);
            }
        }
    }
}
```

Algorithm:

```
static void main()
    1) START
    2) Input sentence as String s.
    3) Count and store number of words in c.
    4) Check if string ends with .?!. If not, display that input is invalid and END.
    5) Remove extra spaces from s.
    6) Store words separately as elements in an array arr.
    7) Input word and position.
    8) Replace position-1 index element's value with blank.
    9) display array as sentence.
    10) END
```

Output:

```
INPUT:  A    MORNING WALK IS A IS BLESSING FOR THE WHOLE DAY.
WORD TO BE DELETED: IS
WORD POSITION IN THE SENTENCE: 6
OUTPUT: A MORNING WALK IS A BLESSING FOR THE WHOLE DAY.
Example 2
INPUT:  AS YOU SOW, SO SO YOU REAP.
WORD TO BE DELETED: SO
WORD POSITION IN THE SENTENCE: 4
OUTPUT: AS YOU SOW, SO YOU REAP.
```

Question 10

An ISBN (International Standard Book Number) is a ten digit code which uniquely identifies a book. The first nine digits represent the Group, Publisher and Title of the book and the last digit is used to check whether ISBN is correct or not.
Each of the first nine digits of the code can take a value between 0 and 9. Sometimes it is necessary to make the last digit equal to ten; this is done by writing the last digit of the code as X. To verify an ISBN, calculate 10 times the first digit, plus 9 times the second digit, plus 8 times the third and so on until we add 1 time the last digit. If the final number leaves no remainder when divided by 11, the code is a valid ISBN.
For example:

1. *02011003311 = 10 x 0 + 9 x 2 + 8 x 0 + 7 x 1 + 6 x 1 + 5 x 0 + 4 x 3 + 3 x 3 + 2 x 1 + 1 x 1 = 55 Since 55 leaves no remainder when divisible by 11, hence it is a valid ISBN.*
2. *007462542X = 10 x 0 + 9 x 0 + 8 x 7 + 7 x 4 + 6 x 6 + 5 x 2 + 4 x 5 + 3 x 4 + 2 x 2 + 1 x 10 = 176 Since 176 leaves no remainder when divided by 11, hence it is a valid ISBN.*
3. *0112112425 = 10 x 0 + 9 x 1 + 8 x 1 + 7 x 2 + 6 x 1 + 5 x 1 + 4 x 1 + 3 x 4 + 2 x 2 + 1 x 5 = 71 Since 71 leaves no remainder when divided by 11, hence it is not a valid ISBN.*

Design a program to accept a ten digit code from the user. For an invalid input, display an appropriate message. Verify the code for its validity in the format specified below:
Test your program with sample data and some random data.

Example 1

INPUT CODE: 0201530821
OUTPUT : SUM = 99
LEAVES NO REMAINDER – VALID ISBN CODE

Example 2

INPUT CODE: 035680324
OUTPUT : INVALID INPUT

Example 3

INPUT CODE: 0231428031
OUTPUT : SUM = 122
LEAVES REMAINDER – INVALID ISBN CODE

Answer 10:

```
import java.io.*;
class ISBN
{
    public static void main(String args[])throws IOException
    {
        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
        System.out.print("Enter a 10 digit code : ");
        String s=br.readLine();

        int len=s.length();
        if(len!=10)
            System.out.println("Output : Invalid Input");
        else
        {
            char ch;
            int dig=0, sum=0, k=10;
            for(int i=0; i<len; i++)
            {
                ch=s.charAt(i);
                if(ch=='X')
                    dig=10;
                else
                    dig=ch-48;
                sum=sum+dig*k;
                k--;
            }

            System.out.println("Output : Sum = "+sum);
            if(sum%11==0)
                System.out.println("Leaves No Remainder - Valid ISBN
Code");
            else
                System.out.println("Leaves Remainder - Invalid ISBN
Code");
        }
    }
}
```

Algorithm:

```
static void main()
    1) START
    2) Input number as string num.
    3) Check if it has 10 digits.If not, display that input is invalid and END.
    4) Convert digits to integers and store the value of their multiplication with 10-theirPosition
       in s.
    5) display s.
    6) Check if s is divisible by 11. If it is, display code is valid. Else display that it is invalid.
    7) END
```

Output:

```
INPUT CODE: 0201530821
OUTPUT: SUM = 99
LEAVES NO REMAINDER – VALID ISBN CODE
```

Question 11

Write a program to declare a square matrix A[][] of order (M X M) where 'M' is the number of rows and the number of columns such that M must be greater than 2 and less than 20. Allow the user to input integers into this matrix. Display appropriate error message for an invalid input. Perform the following tasks:

1. *Display the input matrix.*
2. *Create a mirror image of the inputted matrix.*
3. *Display the mirror image matrix.*

Test your program for the following data and some random data:

Example 1

```
INPUT : M = 3
4     16     12
8      2     14
4      1      3
OUTPUT :
ORIGINAL MATRIX
4     16     12
8      2     14
4      1      3
MIRROR IMAGE MATRIX
12    16     4
14     2     8
3      1     6
```

Example 2

```
INPUT : M = 22
OUTPUT : SIZE OUT OF RANGE
```

Answer 11:

```
import java.io.*
class ImageMatrix
{
    public static void main(String args[])throws IOException
        {
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        System.out.print("Enter the size of the square matrix : ");
        int m = Integer.parseInt(br.readLine());

        if(m>2 && m<20)
        {
            int A[][]=new int[m][m];
            System.out.println("Enter the elements of the Matrix : ");
            for(int i=0;i<m;i++)
            {
                for(int j=0;j<m;j++)
                {
                    A[i][j]=Integer.parseInt(br.readLine());
                }
            }

            System.out.println("The original matrix:");
            for(int i=0;i<m;i++)
            {
                for(int j=0;j<m;j++)
                {
                    System.out.print(A[i][j]+"\t");
                }
                System.out.println();
            }

            int B[][]=new int[m][m];
            for(int i=0;i<m;i++)
            {
                int k=0;
                for(int j=m-1;j>=0;j--)
                {
                    B[i][k]=A[i][j];
                    k++;
                }
            }

            System.out.println("The Mirror Image:");
            for(int i=0;i<m;i++)
            {
                for(int j=0;j<m;j++)
                {
                    System.out.print(B[i][j]+"\t");
                }
                System.out.println();
            }
        }
        else
            System.out.println("Output : Size Out Of Range");
    }
}
```

Algorithm:

static void main()

1) START
2) Input size M from user.
3) Check if it is between 2 and 20. If not, display size invalid and END.
4) Declare 2 2D matrices A and B
5) Take input from user to populate mat at index (i,j). Simultaneously store those values at (i, M-1-j).
6) display A as original matrix.
7) display B as rearranged matrix.
8) END

Output:

```
INPUT: M = 3
4       16      12
8       2       14
4       1       3
OUTPUT:
ORIGINAL MATRIX
```

```
4       16      12
8        2      14
4        1       3
MIRROR IMAGE MATRIX
12      16    4
14       2     8
3        1     4
```

Question 12

A palindrome is a word that may be read the same way in either direction. Accept a sentence in UPPER CASE which is terminated by either ".", "?", or "!". Each word of the sentence is separated by a single blank space.

Perform the following tasks:

1. *Display the count of palindromic words in the sentence.*
2. *Display the palindromic words in the sentence.*

Example of palindromic words:

MADAM, ARORA, NOON

Test your program with the sample data and some random data:

Example 1

```
INPUT : MOM AND DAD ARE COMING AT NOON.
OUTPUT : MOM DAD NOON
NUMBER OF PALINDROMIC WORDS : 3
```

Example 2

```
INPUT : NITIN ARORA USES LIRIL SOAP.
OUTPUT : NITIN ARORA LIRIL
NUMBER OF PALINDROMIC WORDS : 3
```

Example 3

```
INPUT : HOW ARE YOU?
OUTPUT : NO PALINDROMIC WORDS
```

Answer 12:

```
class Palindrome
{
     boolean isPalindrome(String word)
     {
          for(int i=0; i<word.length(); i++)
          {
               if(word.charAt(i) !=
word.charAt(word.length() - i - 1))
                    return false;
          }
          return true;
     }

     void main(String str)
     {
          str = str.toUpperCase();
          String word = "";
          int c=0;

          for(int i=0; i<str.length(); i++)
          {
               char ch = str.charAt(i);
               if(ch == ' ' || ch == '.' || ch  ==
'!')
               {
                    if(isPalindrome(word))
                    {
                         c++;
                         System.out.print(word + "
");
                    }
                    word = "";
               }
               else
                    word = word + ch;
          }

          System.out.print("\n Number of
Palindromic words: " + c);
     }
}
```

Algorithm:

```
static void main()
    1) START
    2) Input sentence as string s.
    3) Declare and set to zero a counter for palin words.
    4) Loop through words of the sentence.
    5) Call isPalindrome() to check if word is palindrome. If it is, add 1 to the counter.
    6) display counter.
    7) END

static boolean isPalindrome(String s)
START
    1. Run a for loop till length of word extracted.
    2. Compare character at index i with character at index word.length() – i – 1 for equality.
    3. If they are same, keep going till all the characters have been tested.
    4. If they are not same, break loop and return false.
    5. Else return True.
    6. END OF ALGORITHM for isPalindrome
```

Output:

```
INPUT : NITIN ARORA USES LIRIL SOAP.
OUTPUT : NITIN ARORA LIRIL
NUMBER OF PALINDROMIC WORDS : 3
```

Question 13

A prime palindrome integer is a positive integer (without leading zeros) which is prime as well as a palindrome. Given two positive integers m and n, where m < n, write a program to determine how many prime-palindrome integers are there in the range between m and n (both inclusive) and output them.
The input contains two positive integers m and n where m < 3000 and n < 3000. Display the number of prime palindrome integers in the specified range along with their values in the format specified below:
Test your program with the sample data and some random data:

<u>Example 1</u>

INPUT:

```
m=100
n = 1000
```

OUTPUT:

```
THE PRIME PALINDROME INTEGERS ARE:
101, 131, 151, 181, 191, 313, 353, 373, 383, 727, 757, 787, 797, 919, 929
FREQUENCY OF PRIME PALINDROME INTEGERS : 15
```

<u>Example 2</u>

INPUT:

```
m = 100
n = 5000
```

OUTPUT: *OUT OF RANGE*

Answer 13:

```
class PrimePal
{
    boolean isPrime(int x)
    {
        int c = -1;
        for(int i=2; i<x; i++)
        {
            if(x%i == 0)
                c++;
        }
        if(c == -1)
            return true;
        else
            return false;
    }

    boolean isPalindrome(int x)
    {
        int temp = x;
        int revnum = 0;
        while(temp != 0)
        {
            int dig = temp%10;
            revnum = revnum*10 + dig;
            temp = temp / 10;
        }
        if(revnum == x)
            return true;
        else
            return false;
    }

    void main(int m, int n)
    {
        int c = 0;
        System.out.print("The prime palindrome integers are:");
        if((m>0 && m<3000) && (n>0 &&   n<3000))
        {
            for(int i=m; i<=n; i++)
            {
                if(isPalindrome(i) && isPrime(i))
                {
                    System.out.print(i + ", ");
                    c++;
                }
            }
            System.out.print("\n Frequency of Prime Palindrome
integers: " + c);
        }
        else
        {
            System.out.print("Out of Range");
            System.exit(0);
        }
    }
}
```

Algorithm:

static void main()
START

1. Input M and N.
2. Declare c as counter for primepalins.
3. Check if M<N and M<3000 and N<3000. If not, display M,N out of range and END.
4. Start a loop with i from M to N.
5. Check if i is prime and paindrome by calling isPrime() and isPalindrome(). If it is, add 1 to c.
6. If c is 1, display i. Else display comma then i.
7. display c.
8. END

static boolean isPalindrome(String s)
START

1. Run a for loop till length of word extracted.
2. Compare character at index i with character at index word.length() – i – 1 for equality.
3. If they are same, keep going till all the characters have been tested.
4. If they are not same, break loop and return false.
5. Else return True.
6. END OF ALGORITHM for isPalindrome

static boolean isPrime(int n)

1. START
2. Start a loop with integer variable x = 2, till x is less than half of n, adding 1 to each iteration.
3. If x divides n with zero remainder, return false.

4. Return true.
5. END OF ALGORITHM FOR isPrime()

Output:

```
INPUT:
m = 100
n = 1000
OUTPUT:
THE PRIME PALINDROME INTEGERS ARE:
101, 131, 151, 181, 191, 313, 353, 373, 383, 727, 757, 787, 797, 919, 929
FREQUENCY OF PRIME PALINDROME INTEGERS : 15
```

Question 14

Write a program to accept a sentence as input. The words in the string are to be separated by a blank. Each word must be in upper case. The sentence is terminated by either '.','!' or '?'. Perform the following tasks:

1. *Obtain the length of the sentence (measured in words)*
2. *Arrange the sentence in alphabetical order of the words.*

Test your program with the sample data and some random data:

Example 1:

INPUT: *NECESSITY IS THE MOTHER OF INVENTION.*

OUTPUT:

Length: 6

Rearranged Sentence:

INVENTION IS MOTHER NECESSITY OF THE

Example 2:

INPUT: *BE GOOD TO OTHERS.*

OUTPUT:

Length: 4

Rearranged Sentence: BE GOOD OTHERS TO

Answer 14:

```
import java.io.*;
public class Q14
{
     public static void main(String args[])throws Exception

      {
        BufferedReader stdin=new BufferedReader(new
InputStreamReader(System.in));
        System.out.print("Enter the sentence:");
        String s=stdin.readLine();
        s=s.toUpperCase();//converts the sentence to uppercase.
        int l=s.length();
        char c=s.charAt(l-1);//extracts the last character
        char ch;
        int pos=0,length=0,w=0;
        String temp;

         if(c=='.'||c=='!'||c=='?')
         {
            for(int i=0;i<l;i++) //computes the length of the
sentence(in words)
            {
                 ch=s.charAt(i);
                 if(ch==' '||ch==c)

                   length++;
            }

            System.out.println("Length:"+length);
            String word[]= new String[length];
            for(int j=0;j<l;j++) //to store each word in array
            {
                ch=s.charAt(j);
                if(ch==' '||ch==c)
                {
                    word[w]=s.substring(pos,j);
                    w++;
                    pos=j+1;
                }
            }
```

```
                for(int k=0;k<length;k++)//arranges the word in
alphabetical order
                {
                  for(int m=0;m<(length-k-1);m++)
                  {
                       if(word[m].compareTo(word[m+1])>0)
                       {
                          temp=word[m];
                          word[m]=word[m+1];
                          word[m+1]=temp;
                       }
                   }
               }

               System.out.print("Rearranged Sentence:");
               for(int r=0;r<length;r++)//prints the rearranged sentence
                   System.out.print(word[r]+" ");
           }
           else
               System.out.print("Invalid input");
    }
}
```

Algorithm:

static void main()
START
Input sentence from user as s.
Store each word in an array ‘word’.
Store no. of words in w.
Sort the array alphabetically.
display array ‘word’.
END

Output:

INPUT: NECESSITY IS THE MOTHER OF INVENTION.
OUTPUT:

```
Length: 6
Rearranged Sentence:
INVENTION IS MOTHER NECESSITY OF THE
```

Question 15

Write a program to declare a matrix A [][] of order (MXN) where 'M' is the number of rows and 'N' is the number of columns such that both M and N must be greater than 2 and less than 20. Allow the user to input integers into this matrix.

Perform the following tasks on the matrix:

1. *Display the input matrix*
2. *Find the maximum and minimum value in the matrix and display them along with their position.*
3. *Sort the elements of the matrix in ascending order using any standard sorting technique and rearrange them in the matrix.*

Output the rearranged matrix.

Example 1

INPUT:

```
M=3
N=4
Entered values: 8,7,9,3,-2,0,4,5,1,3,6,-4
```

OUTPUT:

```
Original matrix:
8  7  9  3
-2  0  4  5
1  3  6  -4
Largest Number: 9
Row: 0
Column: 2
Smallest Number: -4
Row=2
Column=3
Rearranged matrix:
-4  -2  0  1
3  3  4  5
6  7  8  9
```

Answer 15:

```
import java.io.*;
class Sort2D_Method2
{
    public static void main(String args[])throws IOException
    {
        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));

        System.out.print("Enter the no. of  rows: "); //inputting
number of rows
        int m=Integer.parseInt(br.readLine());
        System.out.print("Enter the no. of columns: "); //inputting
number of columns
        int n=Integer.parseInt(br.readLine());
        if (m<2 || m>20 || n<2 || n>20)
        {
            System.out.println("Invalid input");
            System.exit(0);
        }

        int A[][]=new int[m][n]; //creating a 2D array

        /* Inputting the 2D Array */

        for(int i=0;i<m;i++)
        {
            for(int j=0;j<n;j++)
            {
                System.out.print("Enter the elements: ");
                A[i][j]=Integer.parseInt(br.readLine());
            }
        }

        /* Printing the original 2D Array */

        System.out.println("The original array:");
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<n;j++)
            {
                System.out.print(A[i][j]+"\t");
            }
            System.out.println();
        }


        /* Caling fnction to find max and min value */
        MaxMinVal(A);


        /* Sorting the 2D Array */
        int t=0;
        for(int x=0;x<m;x++)
        {
            for(int y=0;y<n;y++)
            {
                for(int i=0;i<m;i++)
                {
                    for(int j=0;j<n;j++)
                    {
                        if(A[i][j]>A[x][y])
                        {
                            t=A[x][y];
                            A[x][y]=A[i][j];
                            A[i][j]=t;
                        }
                    }
                }
            }
        }

        /* Printing the sorted 2D Array */

        System.out.println("The Sorted Array:");
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<n;j++)
            {
                System.out.print(A[i][j]+"\t");
            }
            System.out.println();
        }
    }

    static void MaxMinVal(int A[][])
    {
        int max = 0; int min = A[0][0]; int r1 = 0 ; int r2 = 0; int
c1 = 0; int c2 = 0;
        for(int i=0;i<A.length;i++)
        {
            for(int j=0;j<A[0].length;j++)
            {
                if(A[i][j] > max)
                {
                    max = A[i][j];
                    r1 = i; c1 = j;
                }
                if(A[i][j] < min)
                {
                    min = A[i][j];
                    r2 = i; c2 = j;
                }
            }
        }
        System.out.println("Largest Number: " + max + "\n Row: " + r1
+ "\n Column: " + c1 + "\n Smallest Number: " + min + "\n Row: " + r2
+ "\n Column: " + c2);
    }
}
```

Output:

```
INPUT:
M=3 Entered values: 8,7,9,3,-2,0,3,6,-4
 OUTPUT:
Original matrix:
8  7  9
3  -2  0
3  6  -4
Largest Number: 9
Row: 0
Column: 2
Smallest Number: -4
Row=2
Column=2
Rearranged matrix:
-4 -2  0
 3  3  6
 7  8  9
```

Question 16

Write a program to input a natural number less than 1000 and display it in words. Test your program on the sample data and some random data.
Sample input and output of the program

Examples

```
Input: 29
Output: TWENTY NINE
Input: 17001
Output: OUT OF RANGE
Input: 119
Output: ONE HUNDRED AND NINETEEN
Input: 500
Output: FIVE HUNDRED
```

Answer 16:

```
import java.io.*;
class Num2Word
{
    public static void main(String args[]) throws IOException
       {
        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
        String
ty[]={"","","Twenty","Thirty","Forty","Fifty","Sixty","Seventy","Eight
y","Ninety"};
        String
ten[]={"","Ten","Eleven","Twelve","Thirteen","Fourteen","Fifteen","Six
teen","Seventeen",
                      "Eighteen","Nineteen"};
        String
unit[]={"","One","Two","Three","Four","Five","Six","Seven","Eight","Ni
ne"};
        System.out.print("Enter a Number : ");
        int n=Integer.parseInt(br.readLine());

k        /*checking whether the number is in the range [1-9999] or
not*/
        if(n<1 || n>9999)
            System.out.println("Out of Range");
        else
        {
            int th=n/1000; //finding the digit at thousand's place
            int h=(n/100)%10; //finding the digit at hundred's place
            int t=(n/10)%10; //finding the digit at ten's place
            int u=n%10; //finding the digit at unit's place
            System.out.print("Output = ");

            /*Condition for printing digit at thousand's place, is
that it should not be zero*/
            if(th!=0)
                System.out.print(unit[th]+" Thousand");

            /*Condition for printing digit at hundred's place, is that
it should not be zero*/
            if(h!=0)
                System.out.print(" "+unit[h]+" Hundred");
            /*Condition for printing the word "And"*/
            if((t!=0 || u!=0)&&(th!=0 || h!=0))
                System.out.print(" And");

            /*Condition for printing digit at ten's place*/
            if(t==1) //When digit at ten's place is 1, we have
different words like Ten, Eleven etc.
                System.out.print(" "+ten[u+1]);
            else //if it is not 1 then we print the words following a
normal pattern
                System.out.print(" "+ty[t]+" "+unit[u]);
        }
    }
}
```

Algorithm:

static void main()

1. START
2. Declare and init String arrays a and b as {"", "ONE", "TWO", "THREE", "FOUR", "FIVE", "SIX", "SEVEN", "EIGHT", "NINE", "TEN", "ELEVEN", "TWELVE", "THIRTEEN", "FOURTEEN", "FIFTEEN", "SIXTEEN", "SEVENTEEN", "EIGHTEEN", "NINETEEN"} and {"", "", "TWENTY", "THIRTY", "FOURTY", "FIFTY", "SIXTY", "SEVENTY", "EIGHTY", "NINETY"} respectively.
3. Check if n is between 0 and 1000. If not, END.
4. If n is less than or equal to 19 display value at nth index of a and end.
5. If n/100 is nonzero, display display value at n/100th index of a. display HUNDRED.
6. If n is not perfect multiple of hundred display AND.
7. Set n as n%100.
8. If n is less than or equal to 19 display value at nth index of a else display value at nth index of b.
9. Set n as n%100.
10. display value at nth index of a.
11. END

Output:

```
Input: 500
Output: FIVE HUNDRED
```

Question 17

Encryption is a technique of coding messages to maintain their secrecy. A String array of size 'n' where 'n' is greater than 1 and less than 10, stores single sentences (each sentence ends with a full stop) in each row of the array.
Write a program to accept the size of the array.

Display an appropriate message if the size is not satisfying the given condition.
Define a string array of the inputted size and fill it with sentences row-wise. Change the sentence of the odd rows with an encryption of two characters ahead of the original character.
Also change the sentence of the even rows by storing the sentence in reverse order. Display the encrypted sentences as per the sample data given below.
Test your program on the sample data and some random data.

Example 1

```
INPUT: n = 4
IT IS CLOUDY.
IT MAY RAIN.
THE WEATHER IS FINE.
IT IS COOL.
OUTPUT:
KV KU ENQWFA.
RAIN MAY IT.
VJG YGCVJGT KU HKPG.
COOL IS IT.
```

Answer 17:

```
import java.util.*;
class cipher
{
    static void main()
    {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        if (!(n>1 && n<10))
        {
            System.out.println("INVALID INPUT");
            return;
        }
        String glitch = sc.nextLine();
        String[] arr = new String[n];

        for(int i = 0; i < n; i ++)
        {
            String ns = "";
            arr[i] = sc.nextLine();

            if((i+1)%2==0) //rev even rows
            {   ns = arr[i]; ns = ns.substring(0, ns.length());
                arr[i] = (new StringBuffer(ns).reverse().toString() +
".");
            }
            else  //shift odd rows
            {
                for(int x = 0; x< arr[i].length(); x++)
                {
                    if(!(arr[i].charAt(x) == ' ' || arr[i].charAt(x)
== '.' ))
                    {
                        ns += "" + ( ( (int)arr[i].charAt(x) + 2 -
(int)'A' ) % 26 + (int)'A'  ); //Shifts by 2 using difference in
asii's remainder with 26
                    }
                    else ns +=arr[i].charAt(x);
                }
                arr[i] = ns;
            }
        }
        for(int i = 0; i < n; i ++)
        {
            System.out.println(arr[i]);
        }
    }
}
```

Algorithm:

```
static void main()
START
Input n.
Check if it is between 1 and 10. If not, display invalid input and END.
Input n sentences in an array.
Loop through sentences of array.
If sentence is even,loop through words of array.
Shift characters by 2+ with wrap around ‘z’/’Z’.
If sentence is odd, sort words in reverse.
display array.
END
```

Output:

```
INPUT: n = 4
IT IS CLOUDY.
IT MAY RAIN.
THE WEATHER IS FINE.
IT IS COOL.
OUTPUT:
KV KU ENQWFA.
RAIN MAY IT.
VJG YGCVJGT KU HKPG.
COOL IS IT.
```

Question 18

Design a program which accepts your date of birth in dd mm yyyy format. Check whether the date entered is valid or not.
If it is valid, display "VALID DATE", also compute and display the day number of the year for the date of birth. If it is invalid, display "INVALID DATE" and then terminate the program.
Testing of the program

Example 1

INPUT: Enter your date of birth in dd mm yyyy format
05
01
2010
OUTPUT: VALID DATE
5

Example 2

INPUT: Enter your date of birth in dd mm yyyy format
03
04
2010
OUTPUT: VALID DATE
93

Example 3

INPUT: Enter your date of birth in dd mm yyyy format
34
06
2010
OUTPUT: INVALID DATE

Answer 18:

```
import java.io.*;
    class Program1
    {
    int days[]={31,28,31,30,31,30,31,31,30,31,30,31};
    BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
    int d,m,y;
    public void takeDate() throws Exception
    {
        System.out.println("Enter your date of birth in dd mm yyyy
format:");
        d=Integer.parseInt(br.readLine());
        m=Integer.parseInt(br.readLine());
        y=Integer.parseInt(br.readLine());

        if(m<=0 || m>12)
        {
            System.out.println("INVALID MONTH");
            return;
        }
        else if(d<=0 || d>12)
        {
            System.out.println("INVALID DATE");
            return;
        }

        y=validDate(d,m,y);
        if(y>0)
        {
            System.out.println(" VALID DATE\n"+y);
        }
        else
        {
            System.out.println(" INVALID DATE");
            return;
        }
    }

    int validDate(int d1,int m1,int y1)
    {
        int li,i;
        li=leap(y1);
        if(d1>days[m1-1])
            return 0;
        else
        {
            for(i=0;i< m1-1;i++)
            {
                d1=d1+days[i];
            }
            d1=d1+li;
            return d1;
        }
    }

    int leap(int y)
    {
        if(y%100==0 && y%400==0)
        return 1;
        else if(y%100!=0 && y%4==0)
        return 1;
        else
        return 0;
    }

    public static void main(String args[]) throws Exception
    {
        Program1 ob=new Program1();
        ob.takeDate();
    }
}
```

Algorithm:

static void main()

1. START
2. Declare an array days and init as {0, 31, feb, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}.
3. Input date as ints dd, mm, y.
4. Check if y is leap, if it is add one to feb.
5. Check that dd,mm,y are positive. If not, display INVALID DATE and END.
6. Check that dd is less than value of days at mm th index. If not, display INVALID DATE and END.
7. display VALID DATE.
8. display sum of sum of values of days upto mm-1th index and dd.
9. END

Output:

```
INPUT: Enter your date of birth in dd mm yyyy format
05
01
2010
OUTPUT: VALID DATE
5
```

Question 19

A bank intends to design a program to display the denomination of an input amount, upto 5 digits. The available denomination with the bank are of rupees 1000,500,100,50,20,10,5,2 and 1.
Design a program to accept the amount from the user and display the break-up in descending order of denominations. (i.e. preference should be given to the highest denomination available) along with the total number of notes. [Note: only the denomination used should be displayed]. Also print the amount in words according to the digits.

Example 1

```
INPUT: 14836
OUTPUT: ONE FOUR EIGHT THREE SIX
DENOMINATION:
1000 X 14 =14000
500 X 1  =500
100  X  3  =300
50 X 1  =50
5 X 1  =5
1 X 1  =1
```

Example 2

```
INPUT: 235001
OUTPUT: INVALID AMOUNT
```

Answer 19:

```
class NoteDeno
{

    static void main(double m)
    {
        if(!(m>0&&m<100000))
        {
            System.out.println("INVALID AMOUNT");
        }
        int nn;
        int n = nn = (int) Math.floor(m);
        String numW[] = {"ZERO ", "ONE ", "TWO ", "THREE ", "FOUR ",
"FIVE ", "SIX ", "SEVEN ", "EIGHT ", "NINE "};
        int[] deno = {1000, 500, 100, 50, 20, 10, 5, 2, 1};
        int[] denoC = new int[deno.length];
        String ns = nn + "";
        for(int i = 0; i<ns.length(); i++)
        {
            System.out.print(numW[Integer.parseInt(""+ns.charAt(i))]);
        }
         System.out.println();

        for(int i = 0; i < deno.length; i++)
        {
            denoC[i] = n / deno[i]; //number of particular deno needed
            if(denoC[i] != 0)n = n%deno[i]; //remaining amount
            System.out.println(deno[i] + " X " + denoC[i] + " = " +
(deno[i]*denoC[i]));
        }

    }
}
```

Algorithm:

static void main()

1. START
2. Input double m.
3. Check if it is between 0 and 100000. If not, NED
4. Store it as floored int n.

5. Declare and init numW[] = {"ZERO ", "ONE ", "TWO ", "THREE ", "FOUR ", "FIVE ", "SIX ", "SEVEN ", "EIGHT ", "NINE "} and int[] deno = {1000, 500, 100, 50, 20, 10, 5, 2, 1}.
6. Declare denoC to store number of notes.
7. Convert n to string ns.
8. Loop through characters of ns, parse each as an integer and display value at corresponding index of numW.
9. Start loop from i to deno.length.
10. Set denoC's i th element as n/deno[i].
11. If it is nonzero, set n as n%deno[i].
12. display number of notes needed of denomination being evaluated at this iteration.
13. END

Output:

```
INPUT: 14836
OUTPUT: ONE FOUR EIGHT THREE SIX
DENOMINATION:
1000 X 14 = 14000
500 X 1 = 500
100 X 3 = 300
20 X 1 = 20
10 X 1 = 10
5 X 1 = 5
1 X 1 = 1
```

Question 20

Given the two positive integers p and q, where p < q. Write a program to determine how many kaprekar numbers are there in the range between 'p' and 'q'(both inclusive) and output them.About 'kaprekar' number:
A positive whole number 'n' that has 'd' number of digits is squared and split into 2 pieces, a right hand piece that has 'd' digits and a left hand piece that has remaining 'd' or 'd-1' digits. If sum of the pieces is equal to the number then it's a kaprekar number.

Example 1

```
INPUT:
   p=1
   q=1000
OUTPUT:
   THE KAPREKAR NUMBERS ARE:
   1,9,45,55,99,297,999
   FREQUENCY OF KAPREKAR NUMBERS IS:8
```

Answer 20:

```
class Kaprekar
{
    boolean isKaprekar(int n)
    {
        int num_sqr = n*n;
        int new_num = 0;
        int digits = countDigits(num_sqr);
        if(digits%2==0)
        {
            int power = digits/2;
            int t = (int)Math.pow(10,power);
            int r_num = num_sqr%t;
            num_sqr /= t;
            new_num = r_num + num_sqr;
        }
        else
        {
            if(digits%2!=0)
            {
                int power = (digits/2) + 1;
                int t = (int)Math.pow(10,power);
                int r_num = num_sqr%t;
                num_sqr /= t;
                new_num = r_num + num_sqr;
            }
        }
        if(new_num == n)
            return true;
        else
           return false;
    }

    int countDigits(int num)
    {
        int cnt = 0;
        while(num != 0)
        {
            num /= 10;
            cnt++;
        }
        return cnt;
    }

    void main(int p, int q)
    {
        for(int i = p; i<=q; i++)
        {
            if(isKaprekar(i))
                System.out.println(i + " Kaprekar");
            else
                System.out.println(i + " is not Kaprekar");
        }
    }
}
```

Output:

```
INPUT:
   p=1
   q=1000
OUTPUT:
   THE KAPREKAR NUMBERS ARE:
   1, 9, 45, 55, 99, 297, 703, 999
   FREQUENCY OF KAPREKAR NUMBERS IS: 8
```

Question 21

Input a paragraph containing 'n' number of sentences where (1<=n<=4). The words are to be separated with single blank space and are in upper case. A sentence may be terminated either with a full stop (.) or question mark (?).

Perform the following:

1. *Enter the number of sentences, if it exceeds the limit show a message.*
2. *Find the number of words in the paragraph*
3. *Display the words in ascending order with frequency.*

Example 1

```
INPUT: Enter number of sentences: 1
Enter sentences:
TO BE OR NOT TO BE.
OUTPUT:
Total number of words: 6
WORD       FREQUENCY
OR            1
NOT           1
TO            2
BE            2
```

Example 2

```
INPUT: Enter number of sentences: 3
Enter sentences:
THIS IS A STRING PROGRAM. IS THIS EASY? YES, IT IS.
OUTPUT:
Total number of words: 11
WORD       FREQUENCY
 A            1
 STRING       1
 PROGRAM      1
 EASY         1
 YES          1
 IT           1
 THIS         2
 IS           3
```

Answer 21:

```
import java.util.*;

class Wordcount
{
    static void main()
    {
        int cs; String[] arr;

        //Taking no of sentences
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of sentences: ");
        cs = sc.nextInt(); System.out.println();
        if(cs<1 || cs>4)
        {
            System.out.println("Sentence limit exceeded.");
            return;
        }
        String glitch = sc.nextLine(); //to rectify scanner input
        String s = sc.nextLine();
        StringTokenizer st = new StringTokenizer(s, "?. " ,false);
        arr = new  String[st.countTokens()];
        for(int i = 0; i<arr.length ; i++)
        {
            arr[i] = st.nextToken();
        }

        System.out.println("Total no of words: " + arr.length);
        System.out.format("%-15s%-15s", "WORD", "FREQUENCY");
System.out.println();

        String w; boolean ae; int wf;  // w = word; ae =
alreadyEncountered?; wf = wordFreq
        for(int i = 0; i<arr.length ; i++)
        {
            ae = false;
            w  = arr[i];
            //if word has been already encountered in the sentence
            for(int j = i-1; j>=0 ; j--)
            {
                if(w.equals(arr[j]))
                {
                    ae = true;
                }
            }

            if(!ae)
            {
                wf = getWordFreq(w, arr);
                System.out.format("%-15s%-15s", w, wf);
System.out.println();
            }
        }
    }

    static int getWordFreq(String w, String[] arr)
    {   int c = 0;
        for(int i = 0; i<arr.length ; i++)
        {
            if(w.equals(arr[i])) c++;
        }
        return c;
    }
}
```

Algorithm:

static void main()

1. START
2. Input number cs of sentences.
3. If cs is not within 1 and 4, END.
4. Input sentences.
5. Separate words and store in an array arr.
6. Loop through array.
7. Check if current word has already been encountered, if not continue.
8. Get frequency of word in array using getWordFreq(w, arr)
9. display word and its frequency.
10. END

static int getWordFreq(String w, String[] arr)

1. START
2. Declare and init c as 0.
3. Loop through arr, every time w is found add 1 to c.

4. Return c.
5. END OF ALGORITHM for getWordFreq(w, arr)

Output:

```
INPUT: Enter number of sentences: 1
Enter sentences:
TO BE OR NOT TO BE.
OUTPUT:
Total number of words: 6
WORD       FREQUENCY
OR              1
NOT             1
TO              2
BE              2
```

Question 22

A class Employee contains employee details and another class Salary calculates the employee's net salary. The details of the two classes are given below.
Class name – Employee
Data Members-:

- *empno - to store employee no*
- *empname - to store employee name*
- *empdesign – to store designation*

Member Functions-:

- *Employee( ) - default contructor*
- *Employee(---) - parameterized constructor*
- *void display( ) – to display data members with proper message*

Class name – Salary
Data Members-:

- *basic -to store the basic pay*

Member Functions-:

- *Salary( ) - default constructor*
- *Salary(---) - parameterized constructor to initialize data members of both classes*
- *void calculate( ) - calculates the net salary according to the following rules-:*

DA=10% of basic
HRA=15% of basic
Salary=basic+DA+HRA
PF=8% of Salary
Net Salary= Salary-PF

Display the details of the class Employee and the net salary
Specify the class Employee giving the details of the functions using the concept of inheritance and also specify the class in which object is created and functions are called in the main function

Answer 22:

```
class Employee
{
      int empno;
      String empname;
      String empdesign;

      Employee(){}

      Employee(int en, String empname, String desig)
      {
            empno = en;
            this.empname = empname;
            empdesign = desig;
      }

      void display()
      {
            System.out.println("Employee No. " + en + " Employee name "
+ empname + " Employee designation " + empdesign);
      }
}


class Salary extends Employee
{
      double basic;

      Salary()
      {
            Super();
      }

      Salary(int en, String empname; String desig, double b)
      {
            Super(en, empname, desig);
            basic = b;
      }
```

```
        void calculate()
        {
                double da = 0.1 * basic;
                double hra = 0.15 * basic;
                double salary = basic + da + hra;
                double pf = 0.08 * salary;
                double net_salary = salary - pf;
                System.out.println(en + " " + empname + " " + desig + " " +
net_salary);
        }
}

class Execute
{
        void main()
        {
                Salary obj = new Salary();
                obj = new Salary(478, "Elon", "Engineer", 2000000);
                obj.calculate();
        }
}
```

Question 23

Design a class Prime.
Data Members-:

- *n - to store the number*

Member Functions-:

- *Prime(int) - constructor to initialize n*
- *void check( ) - to check if the no is prime or not by calling a function isPrime(int)*
- *boolean isPrime(int) - returns true/false for prime using recursive technique*

Answer 23:

```
Class Prime
{
     int n;

     Prime(int nn)
     {
          n = nn;
     }

     void check()
     {
          if(isPrime(n))
               System.out.println("Number is prime");
          else
               System.out.println("Number is not prime");
     }

     boolean isPrime(int x)
     {
          int c = -1;
          for(int i=2; i<x; i++)
          {
               if(x%i == 0)
                    c++;
          }

          if(c != -1)
               return true;
          else
               return false;
     }
}
```

Algorithm:

static boolean isPrime(int n)

1. START
2. Start a loop with integer variable x = 2, till x is less than half of n, adding 1 to each iteration.
3. If x divides n with zero remainder, return false.
4. Return true.
5. END OF ALGORITHM FOR isPrime()

Output:

```
INPUT: 7
OUTPUT: Number is prime.
```

Question 24

Design a class Stack-:

Data Members-:

- *int arr[ ] – size 50*
- *int size – store no of elements*
- *int top*

Member Functions-:

- *Stack(int s) – to initialize the size by s and top by -1*
- *void pushvalues(int v) – to fill the stack if it is empty otherwise display message – "overflow"*
- *int popvalues() – to delete value from the stack and return it, if possible otherwise display – "underflow" and return -999*
- *void display()*

Answer 24:

```
Class Stack
{
     int arr[] = new int[50];
     int size;
     int top;

     Stack(int s)
     {
          size = s;
          top = -1;
     }

     void PushValue(int v)
     {
          if(top < size - 1)
               arr[++top] = v;
          else
               System.out.println("Overflow");
     }

     int PopValue()
     {
          if(top != -1)
               return arr[top--];
          else
          {
               System.out.println("Underflow");
               return -999;
          }
     }

     void display()
     {
          for(int i=top; i>=0; i--)
               System.out.println(arr[i]);
     }

}
```

Question 25

Design a class SQ
Data Members-:

- *int arr[ ] – max size 100*
- *int size – store the capacity of arr*
- *int front*
- *int rear*

Member Functions-:

- *SQ(int s) – to initialize the size and front and rear by -1*
- *void addrear(int v) – to add value at rear if possible otherwise display "Queue full at the rear end… OVERFLOW"*
- *int deletefront() – to delete value from front if possible otherwise display "Queue empty…UNDERFLOW"*
- *void display() – display elements of the queue*

Answer 25:

```
class SQ
{
      int arr[] = new int[100];
      int size, front, rear;

      SQ(int s)
      {
            size = s;
            front = rear = -1;
      }

      void addRear(int v)
      {
            if(rear < size – 1)
            {
                  if(front == -1)
                        front = 0;
                  arr[++rear] = v;
            }
            else
                  System.out.println("Queue full at rear end…
      Overflow");
      }

      int delFront()
      {
            if(front <= rear && front != -1)
            {
                  System.out.println("Value deleted " + arr[front]);
                  return arr[front++];
            }
            else
            {
                  System.out.println("Queue empty… underflow");
                  return -999;
            }
      }
```

```
        void display()
        {
             for(int i=front; i<=rear; i++)
                  System.out.println(arr[i] + “ ”);
        }
   }
```

Bibliography

The following sources were helpful while creating this project:

1) Wikipedia
2) ISC Computer Science with Java – Class 11th and 12th
3) GuideForSchools

I am also thankful to my computer science teacher Mr. Akash Gupta for guiding me while I was completing this project.