Design a program which accepts your date of birth in mm,dd,yyyy format. Check whether the date is a valid date or not.

```
import java.util.*;
public class dateValidity{
    public static void main(String args[])
            throws InputMismatchException{
        Scanner scan=new Scanner(System.in);
        System.out.println("Enter your date of birth in dd mm yyyy format : ");
        int d=scan.nextInt();
        int m=scan.nextInt();
        int y=scan.nextInt();
        int dom[]={31,28,31,30,31,30,31,31,30,31,30,31};
        if(y%400==0 || (y%100!=0 && y%4==0))
            dom[1]=29;
        if(d<=dom[m-1])
        {
            System.out.print("VALID DATE ");
            int i,s=0;
            for(i=0;i< m-1;i++)
                s=s+dom[i];

            s+=d;

            System.out.print(s);
        }else{
            System.out.print("INVALID DATE");
        }
    }
}
```

Output:

```
Enter your date of birth in dd mm yyyy format:

06 11 2004

VALID DATE 311

Process finished with exit code 0
```

Write a program in Java to fill the numbers in a circular fashion(anti-clockwise) with natural numbers from 1 to n2, taking 'n' as input. The elements should start filling from the center cell.

```java
import java.util.Scanner;
public class anticlockwiseArray
{
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the size of the matrix: ");
        int n = in.nextInt();
        if (n % 2 == 0) {
            System.out.println("Invalid Input! Size must be an odd number");
            return;
        }
        int val  = 1;
        int arr[][] = new int[n][n];
        int x = n / 2;
        int y = n / 2;
        int d = 1;
        int c = 0;
        int s = 1;
        for (int k = 1; k <= (n - 1); k++) {
            for (int j = 0; j < (k < n - 1 ? 2 : 3); j++) {
                for (int i = 0; i < s; i++) {
                    arr[x][y] = val++;

                    switch (d) {
                        case 0:
                            y = y - 1;
                            break;

                        case 1:
                            x = x + 1;
                            break;

                        case 2:
                            y = y + 1;
                            break;

                        case 3:
                            x = x - 1;
                            break;
                    }
                }

                d = (d + 1) % 4;
            }
            s = s + 1;
        }
        arr[n-1][0] = val;
        System.out.println("Circular Matrix AntiClockwise:");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(arr[i][j] + "\t");
            }
            System.out.println();
        }
    }
}
```

Output:

```
Enter the size of the matrix: 5

Circular Matrix AntiClockwise:

21   20   19   18   17

22   7    6    5    16

23   8    1    4    15

24   9    2    3    14

25   10   11   12   13
```

Design a program to accept the amount from the user and display the break-up in descending order of denominations

```
import java.util.*;
public class bankDenomination{
    public static void main(String args[])
            throws InputMismatchException{

        Scanner scan=new Scanner(System.in);
        int amt;

        System.out.print("Enter a five-digit amount : ");
        amt=scan.nextInt();

        if(amt>99999){
            System.out.println("INVALID AMOUNT.");}
        else{
            int a[]={2000/2,500,100,50,20,10,5,2,1};int i,p,r,b,t;

            p=amt;
            for(i=0;i< a.length;i++){
                t=amt/a[i];
                if(t!=0){
                    System.out.println(a[i]+"X"+t+"="+(t*a[i]));
                    amt=amt%a[i];
                }
            }

            String ones[]={"one","two","three","four","five",
                    "six","seven","eight","nine"};
            r=0;
            while(p>0){
                r=r*10+p%10;
                p/=10;
            }

            while(r>0){
                b=r%10;
                System.out.print(ones[b-1].toUpperCase()+" ");
                r/=10;
            }
        }
    }
}
```

Output:

```
Enter a five-digit amount: 10564

2000X5=10000

500X1=500

50X1=50

10X1=10

2X2=4
```

Write a program to fill the numbers in a circular fashion in an array

```
import java.util.Scanner;
public class clockwiseArray
{
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the size of the matrix: ");
        int n = in.nextInt();

        if (n % 2 == 0) {
            System.out.println("Invalid Input! Size must be an odd number");
            return;
        }

        int val  = 1;
        int arr[][] = new int[n][n];

        int x = n / 2;
        int y = n / 2;
        int d = 0;
        int c = 0;
        int s = 1;

        for (int k = 1; k <= (n - 1); k++) {
            for (int j = 0; j < (k < n - 1 ? 2 : 3); j++) {
                for (int i = 0; i < s; i++) {
                    arr[x][y] = val++;

                    switch (d) {
                        case 0:
                            y = y + 1;
                            break;

                        case 1:
                            x = x + 1;
                            break;

                        case 2:
                            y = y - 1;
                            break;

                        case 3:
                            x = x - 1;
                            break;
                    }
                }

                d = (d + 1) % 4;
            }

            s = s + 1;
        }

        arr[0][n-1] = val;

        System.out.println("Circular Matrix Clockwise:");

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(arr[i][j] + "\t");
            }
            System.out.println();
        }
    }
}
```

Output:

```
Enter the size of the matrix:4

Circular Matrix Clockwise:
1      2      3      4

12     13     14     5

11     16     15     6

10     9      8      7
```

Design a program to accept a day number(between 1 to 366), year (4 digits) from the user to generate and display the corresponding date

```
import java.util.Scanner;
public class dayAndDate
{
    public static boolean isLeapYear(int y) {
        boolean ret = false;

        if (y % 400 == 0) {
            ret = true;
        }
        else if (y % 100 == 0) {
            ret = false;
        }
        else if (y % 4 == 0) {
            ret = true;
        }
        else {
            ret = false;
        }

        return ret;
    }

    public static String computeDate(int day, int year) {
        int monthDays[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
        String monthNames[] = {"JANUARY", "FEBRUARY", "MARCH",
                "APRIL", "MAY", "JUNE",
                "JULY", "AUGUST", "SEPTEMBER",
                "OCTOBER", "NOVEMBER", "DECEMBER"};

        boolean leap = isLeapYear(year);

        if (leap) {
            monthDays[1] = 29;
        }

        int i = 0;
        int daySum = 0;
        for (i = 0; i < monthDays.length; i++) {
            daySum += monthDays[i];
            if (daySum >= day) {
                break;
            }
        }

        int date = day + monthDays[i] - daySum;

        StringBuffer sb = new StringBuffer();
        sb.append(date);
        sb.append("TH ");
        sb.append(monthNames[i]);
        sb.append(", ");
        sb.append(year);

        return sb.toString();
    }

    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.print("DAY NUMBER: ");
        int dayNum = in.nextInt();
        System.out.print("YEAR: ");
        int year = in.nextInt();
        System.out.print("DATE AFTER (N DAYS): ");
        int n = in.nextInt();

        if (dayNum < 1 || dayNum > 366) {
            System.out.println("DAY NUMBER OUT OF RANGE");
            return;
        }

        if (n < 1 || n > 100) {
            System.out.println("DATE AFTER (N DAYS) OUT OF RANGE");
            return;
        }

        String dateStr = computeDate(dayNum, year);

        int nDays = dayNum + n;
        int nYear = year;
        boolean leap = isLeapYear(year);

        if (leap && nDays > 366) {
            nYear = nYear + 1;
            nDays = nDays - 366;
        }
        else if (nDays > 365) {
            nYear = nYear + 1;
            nDays = nDays - 365;
        }

        String nDateStr = computeDate(nDays, nYear);

        System.out.println();
        System.out.println("DATE: " + dateStr);
        System.out.println("DATE AFTER " + n
                + " DAYS: " + nDateStr);
    }
}
```

Output:

```
DAY NUMBER: 255

YEAR: 2018

DATE AFTER (N DAYS): 22


DATE: 12TH SEPTEMBER, 2018

DATE AFTER 22 DAYS: 4TH OCTOBER, 2018
```

Write a program to accept a date in the string format dd/mm/yyyy and accept the name of the day of 1st January of the year. Find the day for the given date.

```
import java.util.*;
public class dayName
{

    public static void main(String args[]) {
        int monthDays[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
        String dayNames[] = {"MONDAY", "TUESDAY", "WEDNESDAY", "THURSDAY",
                "FRIDAY", "SATURDAY", "SUNDAY"};
        Scanner in = new Scanner(System.in);
        System.out.print("Enter Date(dd/mm/yyyy): ");
        String dateStr = in.nextLine();

        StringTokenizer st = new StringTokenizer(dateStr, "/");
        int tokenCount = st.countTokens();
        if (tokenCount <= 0 || tokenCount > 3) {
            System.out.println("Invalid Date");
            return;
        }

        int day = Integer.parseInt(st.nextToken());
        int month = Integer.parseInt(st.nextToken());
        int year = Integer.parseInt(st.nextToken());

        boolean leapYear = isLeapYear(year);
        if (leapYear) {
            monthDays[1] = 29;
        }

        if (month < 1 || month > 12) {
            System.out.println("Invalid Month");
            return;
        }

        if (day < 1 || day > monthDays[month - 1]) {
            System.out.println("Invalid Day");
            return;
        }

        System.out.print("Day on 1st January: ");
        String startDayName = in.nextLine();

        int startDayIdx = -1;
        for (int i = 0; i < dayNames.length; i++) {
            if (dayNames[i].equalsIgnoreCase(startDayName)) {
                startDayIdx = i;
                break;
            }
        }

        if (startDayIdx == -1) {
            System.out.println("Invalid Day Name");
            return;
        }

        //Calculate total days
        int tDays = 0;
        for (int i = 0; i < month - 1; i++) {
            tDays += monthDays[i];
        }

        tDays += day;

        int currDayIdx = tDays % 7 + startDayIdx - 1;
        if (currDayIdx >= 7) {
            currDayIdx -= 7;
        }

        System.out.println("Day on " + dateStr + " : " + dayNames[currDayIdx]);
    }

    public static boolean isLeapYear(int y) {
        boolean ret = false;

        if (y % 400 == 0) {
            ret = true;
        }
        else if (y % 100 == 0) {
            ret = false;
        }
        else if (y % 4 == 0) {
            ret = true;
        }
        else {
            ret = false;
        }

        return ret;
    }
}
```

Output:

```
Enter Date(dd/mm/yyyy): 04/9/1988

Day on 1st January: Thursday

Day on 04/9/1988: Friday
```

Write a program to input two valid dates each comprising of Day(2 digits), Month(2 digits)and Year(4 digits). Calculate the days elapsed between the two dates.

```
import java.util.Scanner;
public class daysBetween
{
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        int d1[] = new int[3];
        int d2[] = new int[3];
        System.out.println("Enter First Date: ");
        System.out.print("Enter Day: ");
        d1[0] = in.nextInt();
        System.out.print("Enter Month: ");
        d1[1] = in.nextInt();
        System.out.print("Enter Year: ");
        d1[2] = in.nextInt();
        System.out.println("Enter Second Date: ");
        System.out.print("Enter Day: ");
        d2[0] = in.nextInt();
        System.out.print("Enter Month: ");
        d2[1] = in.nextInt();
        System.out.print("Enter Year: ");
        d2[2] = in.nextInt();

        int monthDays[] = {31, 28, 31, 30, 31, 30,
                31, 31, 30, 31, 30, 31};

        int n1 = d1[2] * 365 + d1[0];
        int n2 = d2[2] * 365 + d2[0];

        for (int i = 0; i < d1[1] - 1; i++) {
            n1 += monthDays[i];
        }

        for (int i = 0; i < d2[1] - 1; i++) {
            n2 += monthDays[i];
        }

        int y = d1[1] <= 2 ? d1[2] - 1 : d1[2];
        n1 += y / 4 - y / 100 + y / 400;

        y = d2[1] <= 2 ? d2[2] - 1 : d2[2];
        n2 += y / 4 - y / 100 + y / 400;

        int daysElapsed = Math.abs(n2 - n1);
        System.out.println("No of days Elapsed = " + daysElapsed);
    }
}
```

Output:

```
Enter First Date:

Enter Day: 24

Enter Month: 09

Enter Year: 1960

Enter Second Date:

Enter Day: 08

Enter Month: 12

Enter Year: 1852

No of days Elapsed: 39371
```

Write a program to input and store 'n' integers (n>0) in a single subscripted variable and print each number with their frequencies of existence.

```
import java.util.Scanner;
public class frequencyOfDigit
{
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter n: ");
        int n = in.nextInt();

        if (n <= 0) {
            System.out.println("Invalid Input! n should be greater than 0.");
            return;
        }

        int arr[] = new int[n];

        System.out.println("Enter array elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = in.nextInt();
        }
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    int t = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = t;
                }
            }
        }

        System.out.println("Number\tFrequency");
        int count = 0;
        for (int i = 0; i < n - 1; i++) {
            count++;
            if (arr[i] != arr[i + 1]) {
                System.out.println(arr[i] + "\t" + count);
                count = 0;
            }
        }
        count++;
        System.out.println(arr[n - 1] + "\t" + count);
    }
}
```

Output:

```
Number      Frequency

12          4

14          3

16          2

18          2

20          3
```

Write a program to enter ‘n’ natural numbers in a double dimensional array of order m*n. Display the greatest element of the matrix. Replace the elements of the left and right diagonal elements with the greatest element of the matrix.

```
public class greatestDiagonal
{

    static final int MAX = 100;
    static void print(int mat[][], int n, int m)
    {
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < m; j++)
            {
                System.out.print(mat[i][j] + " ");
            }

            System.out.println();
        }
    }

    static void makediagonalgreatest(int mat[][],
                                int n, int m)
    {
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < m; j++)
            {
                if (i == j || (i + j + 1) == n)
                {
                    mat[i][j] = 0;
                }
            }
        }
        print(mat, n, m);
    }
    public static void main(String args[])
    {

        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of rows (m): ");
        int n = sc.nextInt();
        System.out.print("Enter number of columns (n): ");
        int m = sc.nextInt();
        int mat[][] = new int[m][n];
        System.out.println("Enter array elements");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                mat[i][j] = sc.nextInt();
            }
        }
        makediagonalgreatest(mat, m, n);
    }
}
```

Output:

```
Enter number of rows (m): 4

Enter number of columns (n):

Enter array elements

1  2  3  4
12 13 14 5
11 16 15 6
10 9  8  7
Greatest element: 16

Edited Matrix:

16 2 3 16
12 16 16 5
11 16 16 6
16 9 8 16
```

Design a program:
a) To find the duration for which each user logged. Output all records along with the duration
in housrs
b) Output the record of the user who logged for the longest duration. You may assume that no user will login for more than 48 hours.

```
import java.util.Scanner;
public class logInOff
{
    public static void main(String args[]) {
        final int MINS_IN_DAY = 1440;
        final int MINS_IN_HOUR = 60;
        Scanner in = new Scanner(System.in);
        System.out.print("Enter Number of Users: ");
        int n = in.nextInt();
        in.nextLine();

        if (n > 100 || n < 1) {
            System.out.println("Invalid Input!");
            System.out.println("No of users must be between 1 and 100");
            return;
        }

        String records[][] = new String[n][6];

        int monthDays[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

        for (int i = 0; i < n; i++) {
            System.out.println("Enter record of user " + (i+1) + ": ");
            System.out.print("Enter User Identification: ");
            records[i][0] = in.nextLine();
            System.out.print("Enter Login Time(hh:mm): ");
            records[i][1] = in.nextLine();
            System.out.print("Enter Login Date(dd-mm): ");
            records[i][2] = in.nextLine();
            System.out.print("Enter Logout Time(hh:mm): ");
            records[i][3] = in.nextLine();
            System.out.print("Enter Logout Date(dd-mm): ");
            records[i][4] = in.nextLine();
        }
        System.out.println();

        int longIdx = 0;
        int longDuration = 0;
        for (int i = 0; i < n; i++) {
            int duration = 0;
            int tempIdx = records[i][1].indexOf(':');
            int loginHr = Integer.parseInt(records[i][1].substring(0, tempIdx));
            int loginMin = Integer.parseInt(records[i][1].substring(tempIdx + 1));
            tempIdx = records[i][3].indexOf(':');
            int logoutHr = Integer.parseInt(records[i][3].substring(0, tempIdx));
            int logoutMin = Integer.parseInt(records[i][3].substring(tempIdx + 1));
            int m1 = loginHr * MINS_IN_HOUR + loginMin;
            int m2 = logoutHr * MINS_IN_HOUR + logoutMin;

            //If login & logout is on the same day
            if (records[i][2].equals(records[i][4])) {
                duration = m2 - m1;
            }
            else {
                int daysDiff = 0;
                tempIdx = records[i][2].indexOf('-');
                int loginDay = Integer.parseInt(records[i][2].substring(0, tempIdx));
                int loginMonth = Integer.parseInt(records[i][2].substring(tempIdx +
1));
                tempIdx = records[i][4].indexOf('-');
                int logoutDay = Integer.parseInt(records[i][4].substring(0, tempIdx));
                int logoutMonth = Integer.parseInt(records[i][4].substring(tempIdx +
1));
                //If login & logout is in the same month
                if (loginMonth == logoutMonth) {
                    daysDiff = logoutDay - loginDay - 1;
                }
                else {
                    daysDiff = monthDays[loginMonth - 1] - loginDay + logoutDay - 1;
                }

                duration = (MINS_IN_DAY - m1) + m2 + daysDiff * MINS_IN_DAY;
            }

            if (duration > longDuration) {
                longDuration = duration;
                longIdx = i;
            }

            int durHr = duration / 60;
            int durMin = duration % 60;
            records[i][5] = (durHr == 0 ? "00" : durHr)
                    + ":"
                    + (durMin == 0 ? "00" : durMin);
        }

        System.out.println("User\t\tLogin\t\tLogout\t\tDuration");
        System.out.println("Identification\tTime & Date\tTime & Date\tHours:Minutes");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < 6; j++) {
                System.out.print(records[i][j] + "\t");
            }
            System.out.println();
        }

        System.out.println();
        System.out.println("The user who logged in for longest duration:");
        for (int j = 0; j < 6; j++) {
            System.out.print(records[longIdx][j] + "\t");
        }
    }
}
```

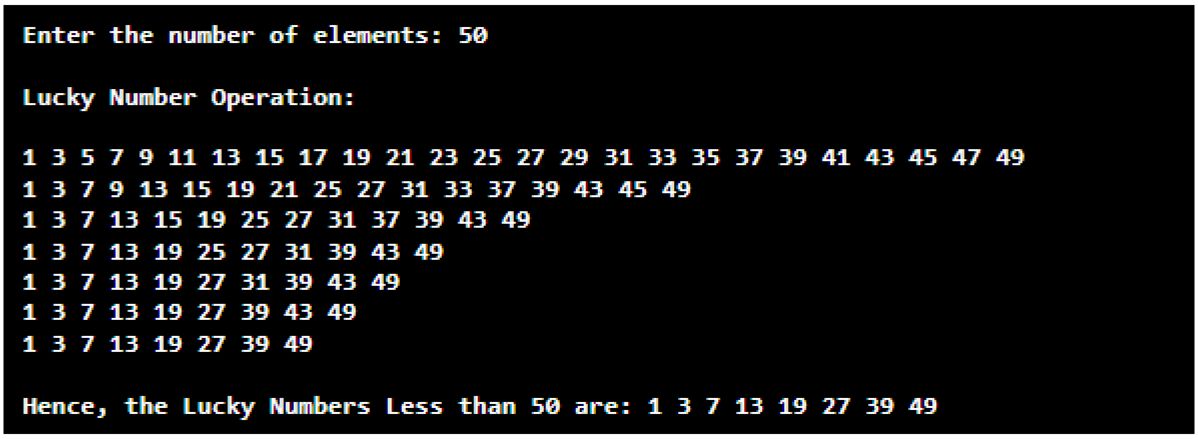
Output:

```
User Identification   Login Time and Date   Logout Time and Date  Duration

149                   20:10 20-12           2:50  21-12           6:40

173                   12:30 20-12           12:30 21-12           24:00

142                   16:20 20-12           16:30 20-12           00:10
```

Write a program to generate and print the Lucky numbers less than a given natural number where 'n' is <=50.

```
import java.util.*;
public class luckyNumber
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the number of elements: ");
        int n=sc.nextInt();
        if(n>50)
        {
            System.out.println("invalid input, range too large");
            System.exit(1);
        }
        else
        {
            int arr[] = new int[n];
            int elements = n;
            for (int i = 0; i < n; i++) {
                arr[i] = i + 1;
            }
            int del = 1;
            System.out.println("\nLucky Number Operation:\n");
            while (del < n) {
                for (int i = del; i < n; i += del) {
                    for (int j = i; j < n - 1; j++) {
                        arr[j] = arr[j + 1];
                    }
                    n--;
                }
                del++;
                for (int i = 0; i < n; i++) {
                    System.out.print(arr[i] + " ");
                }
                System.out.println();
            }
            System.out.print("\nHence, the Lucky Numbers Less than " + elements + "
are: ");
            for (int i = 0; i < n; i++) {
                System.out.print(arr[i] + " ");
            }
        }
    }
}
```

Output:

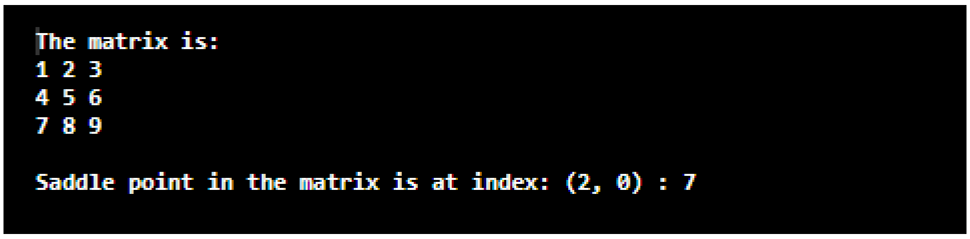


```
Enter the number of elements: 50

Lucky Number Operation:

1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
1 3 7 9 13 15 19 21 25 27 31 33 37 39 43 45 49
1 3 7 13 15 19 25 27 31 37 39 43 49
1 3 7 13 19 25 27 31 39 43 49
1 3 7 13 19 27 31 39 43 49
1 3 7 13 19 27 39 43 49
1 3 7 13 19 27 39 49

Hence, the Lucky Numbers Less than 50 are: 1 3 7 13 19 27 39 49
```

Write a program in java to find the saddle point of a matrix

```
import java.util.Scanner;
public class saddlePoint
{
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the size of the matrix: ");
        int n = in.nextInt();

        if (n >= 20) {
            System.out.print("Size is out of range");
            return;
        }

        int a[][] = new int[n][n];

        System.out.println("Enter elements of the matrix: ");
        for (int i = 0; i < n; i++) {
            System.out.println("Enter Row "+ (i+1) + " :");
            for (int j = 0; j < n; j++) {
                a[i][j] = in.nextInt();
                if (a[i][j] < 0) {
                    System.out.print("Invalid Input");
                    return;
                }
            }
        }

        System.out.println("Matrix A[ ][ ]");
        printMatrix(a, n, n);

        //Find the Saddle Point
        boolean found = false;
        for (int i = 0; i < n; i++) {
            int rMin = a[i][0];
            int cIdx = 0;

            for (int j = 1; j < n; j++) {
                if (rMin > a[i][j]) {
                    rMin = a[i][j];
                    cIdx = j;
                }
            }

            int k = 0;
            for (k = 0; k < n; k++) {
                if (rMin < a[k][cIdx]) {
                    break;
                }
            }

            if (k == n) {
                found = true;
                System.out.println("Saddle Point = " + rMin);
                break;
            }
        }

        if (!found) {
            System.out.println("No Saddle Point");
        }

        //Sort Left Diagonal with Insertion Sort
        for (int i = 1; i < n; i++) {
            int key = a[i][i];
            int j = i - 1;
            while (j >= 0 && a[j][j] > key) {
                a[j + 1][j + 1] = a[j][j];
                j--;
            }

            a[j + 1][j + 1] = key;
        }

        System.out.println("Matrix after sorting the Principal diagonal:");
        printMatrix(a, n, n);
    }

    public static void printMatrix(int arr[][], int m, int n) {
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(arr[i][j] + "\t");
            }
            System.out.println();
        }
    }
}
```

Output:



```
The matrix is:
1 2 3
4 5 6
7 8 9

Saddle point in the matrix is at index: (2, 0) : 7
```

Write a program in Java to enter natural numbers in a double dimensional array m * n (where m is the number of rows and n is the number of columns). Shift the elements of the 4th column to the 1st column, the elements of the first column to the 2nd column and so on. Display the new matrix.

```java
import java.util.Scanner;
public class shiftElements
{
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter number of rows (m): ");
        int m = in.nextInt();
        System.out.print("Enter number of columns (n): ");
        int n = in.nextInt();

        int arr[][] = new int[m][n];
        int newArr[][] = new int[m][n];

        System.out.println("Enter array elements");
        for (int i = 0; i < m; i++) {
            System.out.println("Enter Row "+ (i+1) + " :");
            for (int j = 0; j < n; j++) {
                arr[i][j] = in.nextInt();
            }
        }

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

        for (int j = 0; j < n; j++) {
            int col = j + 1;
            if (col == n) {
                col = 0;
            }
            for (int i = 0; i < m; i++) {
                newArr[i][col] = arr[i][j];
            }
        }

        System.out.println("New Shifted Array:");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(newArr[i][j] + "\t");
            }
            System.out.println();
        }
    }
}
```

Output:

```
Enter number of rows (m): 4

Enter number of columns (n): 4

Enter array elements

11    16    7     4

8     10    9     18

9     8     12    15

14    15    13    6


New Shifted Array:

4     11    16    7

18    8     10    9

15    9     8     12

6     14    15    13
```

Write a program to enter a number and check if it is a Smith number or not.

```
import java.util.Scanner;
public class smithNumber
{
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter number: ");
        int n = in.nextInt();

        if (n <= 0) {
            System.out.println(n + " is not a Smith Number.");
            return;
        }

        boolean isComposite = false;
        for (int i = 2; i < n; i++) {
            if (n % i == 0) {
                isComposite = true;
                break;
            }
        }

        if (isComposite && n != 1) {
            int sumDigits = 0;
            int t = n;
            while (t != 0) {
                int d = t % 10;
                sumDigits += d;
                t /= 10;
            }

            int sumPrimeDigits = 0;
            t = n;
            for(int i = 2; i < t; i++) {
                while(t % i == 0) {
                    t /= i;
                    int temp = i;
                    while (temp != 0) {
                        int d = temp % 10;
                        sumPrimeDigits += d;
                        temp /= 10;
                    }
                }
            }

            if(t > 2) {
                while (t != 0) {
                    int d = t % 10;
                    sumPrimeDigits += d;
                    t /= 10;
                }
            }

            if (sumPrimeDigits == sumDigits)
                System.out.println(n + " is a Smith Number.");
            else
                System.out.println(n + " is not a Smith Number.");
        }
        else {
            System.out.println(n + " is not a Smith Number.");
        }
    }
}
```

Output:

```
Enter number:
22 is a smith number
```

Write a program to enter 'n' natural numbers in a double dimensional array of order m*n. Sort the elements of the principal diagonal elements in descending order. Display the new matrix

```
import java.util.Scanner;
public class sortDiagonal{
    static void sortDiagonal(int a[][], int M, int N)
    {
        for (int i = 0; i < M; i++) {
            int sm = a[i][i];
            int pos = i;
            for (int j = i + 1; j < N; j++) {
                if (sm > a[j][j]) {
                    sm = a[j][j];
                    pos = j;
                }
            }
            swap(a, i, i, pos, pos);
        }
        for (int i = 0; i < M; i++) {
            for (int j = 0; j < N; j++)
                System.out.print(a[i][j]+ " ");
            System.out.println();
        }
    }

    static void swap(int[][] a, int i, int i2, int pos, int pos2) {
        int temp = a[i][i2];
        a[i][i2] = a[pos][pos2];
        a[pos][pos2] = temp;
    }
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of rows (m): ");
        int n = sc.nextInt();
        System.out.print("Enter number of columns (n): ");
        int m = sc.nextInt();
        int a[][] = new int[m][n];
        System.out.println("Enter array elements");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                a[i][j] = sc.nextInt();
            }
        }
        sortDiagonal(a, m, n);
    }
}
```

Output:

```
Enter number of rows (m): 2

Enter number of columns (n):2

Enter array elements

4 2

3 1

Sorted Matrix:

1 2

3 4
```

Write a program in Java to display all Triangular numbers from 3 to n, taking the value of 'n' as input.

```
import java.util.Scanner;
public class triangularNumber
{
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter number: ");
        int n = in.nextInt();

        if (n < 3) {
            System.out.println("Value of n should be greater than or equal to 3");
            return;
        }

        System.out.println("Triangular Numbers from 3 to "
                + n + ":");
        int sum = 3;
        for (int i = 3; sum <= n; i++) {
            System.out.println(sum);
            sum += i;
        }
    }
}
```

Output:

```
Enter number: 10

Triangular Numbers from 3 to 10 are: 3 5 10
```

Write a program to determine the number of Unique digits in the range between m and n(both inclusive). Also, display the Unique digits. A unique-digit integer is a positive integer(without leading zeroes) with no duplicate digits

```java
import java.util.Scanner;
public class uniqueDigitInteger
{
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter m: ");
        int m = in.nextInt();

        if (m < 1 || m > 30000) {
            System.out.println("Value of m should be between 1 and 30000");
            return;
        }

        System.out.print("Enter n: ");
        int n = in.nextInt();

        if (n < 1 || n > 30000) {
            System.out.println("Value of n should be between 1 and 30000");
            return;
        }

        if (m > n) {
            System.out.println("Value of m should be less than n");
            return;
        }

        System.out.println("The Unique-Digit integers are:");
        int count = 0;
        for (int i = m; i <= n; i++) {
            int num = i;
            boolean visited[] = new boolean[10];
            boolean isUnique = true;

            while (num != 0) {
                int d = num % 10;
                if (visited[d]) {
                    isUnique = false;
                    break;
                }
                visited[d] = true;
                num /= 10;
            }

            if (isUnique) {
                count++;
                System.out.print(i + " ");
            }
        }
        System.out.println();
        System.out.println("Frequency of unique-digit integers is: " + count);
    }
}
```

Output:

```
Enter m: 100
Enter n: 120
The Unique-Digit integers are:
102, 103, 104, 105, 106, 107, 108, 109, 120.
Frequency of unique-digit integers is: 9
```

Write a program in Java to read n (2<=n <=10) and the value stored in these n*n cells and the display iff the grid represents a wondrous square.

```
import java.util.Scanner;
public class wondrousSquare
{
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter n: ");
        int n = in.nextInt();

        if (n < 2 || n > 10) {
            System.out.println("Invalid value of n!");
            return;
        }

        int a[][] = new int[n][n];

        System.out.println("Enter elements of the matrix: ");
        for (int i = 0; i < n; i++) {
            System.out.println("Enter Row "+ (i+1) + " :");
            for (int j = 0; j < n; j++) {
                a[i][j] = in.nextInt();
            }
        }

        System.out.println("The Matrix is:");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(a[i][j] + "\t");
            }
            System.out.println();
        }

        //Check Wondrous
        int nSq = n * n;
        double validSum = 0.5 * n * (nSq + 1);
        boolean wondrous = isWondrous(a);

        if (wondrous) {
            System.out.println("Yes, it represents a wondrous square");
        }
        else {
            System.out.println("Not a wondrous square");
        }

        //Print Prime Numbers
        printPrime(a);
    }

    public static boolean isWondrous(int arr[][]) {
        int n = arr.length;
        int nSq = n * n;
        double validSum = 0.5 * n * (nSq + 1);

        /*
         * seenArr is used to check that
         * numbers are not repeated
         */
        boolean seenArr[] = new boolean[nSq];

        for (int i = 0; i < n; i++) {

            int rSum = 0, cSum = 0;
            for (int j = 0; j < n; j++) {
                if (arr[i][j] < 1 || arr[i][j] > nSq) {
                    return false;
                }

                //Number is not distinct
                if (seenArr[arr[i][j] - 1]) {
                    return false;
                }

                seenArr[arr[i][j] - 1] = true;

                rSum += arr[i][j];
                cSum += arr[j][i];
            }

            if (rSum != validSum || cSum != validSum) {
                return false;
            }
        }

        return true;
    }

    public static void printPrime(int arr[][]) {

        int n = arr.length;

        System.out.println("Prime\tRow Index\tColumn Index");

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (isPrime(arr[i][j])) {
                    System.out.println(arr[i][j] + "\t" + i + "\t\t" + j);
                }
            }
        }
    }

    public static boolean isPrime(int num) {
        int c = 0;

        for (int i = 1; i <= num; i++) {
            if (num % i == 0) {
                c++;
            }
        }

        return c == 2;
    }
}
```

Output:

```
Yes, it represents a wondrous square
```

Write a program in java to enter a string in mixed case and arrange all of the letters of the string such that the lowercase characters are followed by the uppercase characters

```java
import java.util.Scanner;
public class mixedCase {
    public static void main(String[] args) {
        // Create a Scanner object to read input from the user
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter a mixed-case string
        System.out.print("Enter a mixed-case string: ");
        String str = scanner.nextLine();

        // Initialize empty strings to hold the lowercase and uppercase characters
        String lowercase = "";
        String uppercase = "";

        // Iterate through each character in the string
        for (int i = 0; i < str.length(); i++) {
            char c = str.charAt(i);
            // Check if the character is lowercase or uppercase
            if (Character.isLowerCase(c)) {
                lowercase += c;
            } else if (Character.isUpperCase(c)) {
                uppercase += c;
            }
        }

        // Concatenate the lowercase and uppercase strings and print the result
        String sorted = lowercase + uppercase;
        System.out.println("Sorted string: " + sorted);
    }
}
```

Output:

```
Enter a mixed-case string: ComputerScience

Sorted string: omputercienceCS
```

Write a program in java to check if a word is an anagram or not
import java.util.Scanner;

```java
public class anagram
{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s1="";
        String s2="";
        char ch1;
        char ch2;
        int c = 0;
        System.out.println("Enter a word");
        s1=sc.next();
        System.out.println("Enter another word");
        s2=sc.next();
        if(s1.length()==s2.length())
        {
            for(int i=0; i<s1.length(); i++)
            {
                ch1 = s1.charAt(i);
                for(int j=0; j<s2.length(); j++)
                {
                    ch2 = s2.charAt(j);
                    if(ch2==ch1)
                    {
                        c++;
                        break;
                    }
                }
            }
            if(c==s1.length())
            {
                System.out.println("Anagram");
            }
            else
            {
                System.out.println("Not Anagram");
            }
        }
        else
        {
            System.out.println("Words are not of the same length, please try again.");
            System.exit(1);
        }
    }
}
```

Output:

```
Enter a word: cat

Enter another word: act

Anagram
```

Write a program in java to check if a number is a sphenic number or not
import java.util.Scanner;

```
public class sphenic
{
    public static void main(String[] args)
    {
        int num;
        int count=0;
        int prod=1;
        int n;
        int j=1;
        int i=1;
        int rem2=0;
        int rem=0;
        int x;
        int c=0;
        String factors="";
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a number");
        num = sc.nextInt();
        n = num;
        for(i=1; i<=num;i++)
        {
             rem = num%i;
             if(rem==0)
             {
                 for(j=1; j<=i; j++)
                 {
                     rem2 = i%j;
                     if(rem2==0)
                         count++;
                     else
                         break;
                     if(count==2)
                     {
                         prod *= j;
                         factors = factors + prod +",";
                     }
                     else
                         break;
                 }
             }
             else
                 break;
        }
        if(prod==n) {
            System.out.println("sphenic number");
            System.out.println("prime factors: "+ factors);
        }
        else
            System.out.println("not sphenic");
    }
}
```

Output:

```
Enter a number: 42

Sphenic number
```

Write a program in java to calculate and print the frequency of the characters in an entered string.

```
import java.util.Scanner;
public class charFrequency
{
    public static void main(String args[])
    {
        int ci, i, j, k, l=0;
        String str, str1;
        char c, ch;
        Scanner scan = new Scanner(System.in);

        System.out.print("Enter a String : ");
        str=scan.nextLine();

        i=str.length();
        for(c='A'; c<='z'; c++)
        {
            k=0;
            for(j=0; j<i; j++)
            {
                ch = str.charAt(j);
                if(ch == c)
                {
                    k++;
                }
            }
            if(k>0)
            {
                System.out.println("The character " + c + " has occurred for " + k + "
times");
            }
        }
    }
}
```

Output:

```
Enter a string: bookkeeper

The character k has occurred for 2 times
```

Write a program in java to reverse a word

```
import java.io.*;
import java.util.*;
class revstr
{
    public void main()throws IOException
    {
        Scanner sc=new Scanner(System.in);
        String str="",word;
        int l,l1,i;
        String a[]=new String[20];
        System.out.println("enter string");
        str=sc.nextLine();
        StringTokenizer data=new StringTokenizer(str);
        l=data.countTokens();
        for(i=0;i<l;i++)
        {
            word=data.nextToken();
            a[i]=word;
        }
        for(i=l-1;i>=0;i--)
        {
            System.out.print(a[i]+" ");
        }
    }
}
```

Output:

```
Enter string: hello

olleh
```

Write a program in java using the deque data structure

```
import java.io.*;
class dequeue
{
    int arr[]=new int[10];
    int f,r;
    int i,n;
    String str;
    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
    dequeue()
    {
        f=-1;
        r=-1;
    }
    public void push()throws IOException
    {
        if(r==9)
        {
            System.out.println("Queue Overflow");
            return;
        }
        System.out.println("Specify the location(front or rear):");
        str=br.readLine().toLowerCase();
        System.out.println("Enter the value to insert:");
        n=Integer.parseInt(br.readLine());
        if(f==-1)
        {
            arr[++f]=n;
            r=0;
        }
        else if(str.charAt(0)=='f')
        {
            for(i=r+1;i>f;i--)
                arr[i]=arr[i-1];
            arr[i]=n;
            r++;
        }
        else
        {
            arr[++r]=n;
        }
    }
    public void display()
    {
        if(f==-1)
            return;
        for(i=f;i<=r;i++)
            System.out.println(""+arr[i]);
    }
    public void pop()throws IOException
    {
        if(f==-1)
        {
            System.out.println("Queue Overflow");
            return;
        }
        System.out.println("Specify the location(front  or rear):");
        str=br.readLine().toLowerCase();
        if(f==r)
        {
            f=-1;
            r=-1;
        }
        else if(str.charAt(0)=='f')
        {
            f++;
        }
        else
        {
            r--;
        }
    }
    public static void main(String args[])throws IOException
    {
        char op;
        BufferedReader br;
        dequeue ob=new dequeue();
        while(true)
        {
            br=new BufferedReader(new InputStreamReader(System.in));
            System.out.println("\nPress'P' for Push,'D' for Pop and 'Q' for Quit:");
            op=(char)br.read();

            if(op=='p' || op=='P')
                ob.push();
            else if(op=='d' || op=='D')
                ob.pop();
            ob.display();
            if(op=='q' || op=='Q')
                break;
            br=null;
        }
    }
}
```

Output:

```
Queue overflow
```

Write a program in java to convert a given time in numbers to words

```
import java.io.*;
public class timeinwords
{

    public static void main(String args[]) throws IOException
    {
        InputStreamReader read = new InputStreamReader(System.in);
        BufferedReader x=new BufferedReader (read);
        System.out.println("enter hours");
        int h=Integer.parseInt(x.readLine());
        System.out.println("enter minutes");
        int m=Integer.parseInt(x.readLine());
        if((h>=1&&h<=12)&&(m>=0&&m<=59))
        {
            String word[]={"","one ","two ","three ","four ","five ","six ","seven
","eight ","nine ","ten "," eleven "," twelwe ","
thirteen","fourteen","fifteen","sixteen","seventeen","eighteen","nineteen","twenty","tw
enty one","twenty two","twenty three","twenty four","twenty five","twenty six","twenty
seven","twenty eight","twenty nine"};
            String plu,a;
            if(m==1||m==59)
                plu="minute";
            else
                plu="minutes";
            if(h==12)
                a=word[1];
            else
                a=word[h+1];
            System.out.print("output: \n "+h+":"+m+"-----");
            if(m==0)
                System.out.println(word[h]+"o'clock");
            else  if(m==15)
                System.out.println("quarter past"+word[h]);
            else  if(m==30)
                System.out.println("half past"+word[h]);
            else  if(m==45)
                System.out.println("quarter to"+a);
            else  if(m<30)
                System.out.println(word[m]+" "+plu+" past+words[h] ");
            else
                System.out.println(word[60-m]+" "+plu+" to "+a);
        }
        else
            System.out.println("invalid input!");
    }
}
```

Output:

```
Enter hours 4

Enter minutes 0

4 o'clock
```

Write a program to check if a number is a unique number

```
import java.util.*;
public class UniqueNumber
{
    // instance variables -
    int n;
    String s;
    int flag,i, j,l;
    /**
     * Constructor for objects of class UniqueNumber
     */
    public UniqueNumber()
    {
        // initialise instance variables
        n = 0;
        flag=0;
        s=" ";
    }
    public void input() // to input an integer
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter a number :");

        n=sc.nextInt();
    }
    public void calculate()
    {
        s= Integer.toString(n); // converting the given number into String datatype
// Int datatype is converting to String,
// String helps to extract the character by charAt() and check each character is
matching with the rest.
        l=s.length();
//Loops to check the digits are repeated
        for(i=0;i<l-1;i++)
        {
            for(j=i+1;j<l;j++)
            {
                if(s.charAt(i)==s.charAt(j)) // if any digits are repeated, then it is
not a UniqueNumber
                { flag=1;
                    break;
                }
            }
        }
    }
    public void display()
    {
        calculate(); //invoke calculate()
        if(flag ==0)
            System.out.println(" It is an UniqueNumber");
        else
            System.out.println(" It is not an UniqueNumber");
    }
    public static void main(String args[]) throws Exception
    {
        UniqueNumber un=new UniqueNumber(); //creating object
        un.input();  //calling input() to main()
        un.display(); //calling display() to main()
    }
}
```

Output:

```
Enter a number: 145

It is an UniqueNumber
```

Write a program to perform Binary Search

```
import java.util.*;
public class binsrch
{

    static void BubbleSort(int num[])  //Sorting
    {
        int len=num.length;
        int i,j, temp;
        for(i=0;i<len;i++)
        {
            for( j=0;j<len-i-1;j++)
            {
                if(num[j]>num[j+1])
                {
                    temp=num[j];
                    num[j]=num[j+1];
                    num[j+1]=temp;

                }
            }
        }
        //   System.out.println("Sorted Elements:");
        for(int k=0;k<len;k++)
            System.out.print(num[k]+"\t");
        System.out.println();
    }
    static void BinSearch(int num[],int k)  //Binary Search
    {
        int l=0,u=num.length-1;
        int m=0, found=0;
        while(l<=u)
        {
            m= (l+u)/2;
            if( k > num[m])
                l =m+1;
            else if( k< num[m])
                u=m-1;
            else
            {
                found =1;
                break;

            }

        }
        if(found==1)
            System.out.println(" Element Present at"+" "+ (m+1) +" " +"after sorting");

        else
            System.out.println("Not Present");
    }
    public static void main(String [] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println( "Enter the total number of elements in the array:");
        int n=sc.nextInt();
        int a[]=new int[n];
        System.out.println( "Enter the elements:");
        for(int i=0;i<n;i++)
        {
            a[i]=sc.nextInt();
        }
        System.out.println("Entered Elements");
        for(int i=0;i<n;i++)
        {

            System.out.print(a[i]+"\t");

        }
        System.out.println();



        System.out.println("Enter element to search");
        int ky=sc.nextInt();
        BinSearch(a, ky); // invoking Binary search method
    }

}
```

Output:

```
Enter the total number of elements in the array: 4

Enter the elements: 3 8 6 0

Enter Element to search: 8

Element Present at 2 after sorting
```

Write a program in java to check if a number is a fascinating number

```java
import java.util.Scanner;

public class FascinatingNumber
{
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the number to check: ");
        int num = in.nextInt();

        if (num < 100) {
            System.out.println(num + " is not a Fascinating Number");
            return;
        }

        int num2 = num * 2;
        int num3 = num * 3;

        boolean isFascinating = true;
        String str = "" + num + num2 + num3;
        for (char i = '1'; i <= '9'; i++) {
            int idx1 = str.indexOf(i);
            int idx2 = str.lastIndexOf(i);
            if (idx1 == -1 || idx1 != idx2) {
                isFascinating = false;
                break;
            }
        }

        if (isFascinating)
            System.out.println(num + " is a Fascinating Number");
        else
            System.out.println(num + " is not a Fascinating Number");
    }
}
```

Output:

```
Enter the number to check: 327

327 is a Fascinating Number.
```

Write a program in java to check if a number is a bouncy number

```
import java.util.Scanner;

public class BouncyNumber
{
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int n = in.nextInt();

        if (n < 100) {
            System.out.println(n + " is not a Bouncy Number.");
            return;
        }

        int t = n;
        boolean isIncreasing = true, isDecreasing = true;

        int prev = t % 10;
        while (t != 0) {
            int d = t % 10;
            if (d > prev) {
                isIncreasing = false;
                break;
            }
            prev = d;
            t /= 10;
        }

        t = n;
        prev = t % 10;
        while (t != 0) {
            int d = t % 10;
            if (d < prev) {
                isDecreasing = false;
                break;
            }
            prev = d;
            t /= 10;
        }

        if (!isIncreasing && !isDecreasing)
            System.out.println(n + " is a Bouncy Number.");
        else
            System.out.println(n + " is not a Bouncy Number.");
    }
}
```

Output:

```
Enter a number: 1441

1441 is a Bouncy Number.
```

Write a program to check if a number is an emirp number

```
import java.util.*;

class Emirp
{
    int n, rev,f;
    Emirp(int nn)
    {
        n=nn;
        rev=0;
        f=2;
    }
    int isprime(int x)
    {
        if(n==x)
            return 1;
        else if (n%x==0 || n==1)
            return 0;
        else
            return isprime(x+1);
    }
    void isEmirp()
    {
        int x=n;
        while(x!=0)
        {
            rev=rev*10 + x%10;
            x=x/10;

        }
        int ans1=isprime(f);
        n=rev;
        f=2;
        int ans2=isprime(f);
        if(ans1==1 && ans2==1)
            System.out.println(n+"is an emirp no");
        else
            System.out.println(n+"is not an emirp no.");

    }
    public static void main(String[] args)
    {
        int a;
        Scanner sc= new Scanner(System.in);
        System.out.println("Enter the number:");
        a=sc.nextInt();
        Emirp emp=new Emirp(a);
        emp.isEmirp();

    }
}
```

Output:

```
Enter the number: 13

13 is an emirp no
```

Write a program in java to accept a string and pass it to a recursive function that displays all its upper case letters. If no upper case letter is present in the string then it should display a message "No upper case letter".

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a string: ");
        String str = sc.nextLine();

        boolean found = false;
        printUpperCaseLetters(str, 0, found);

        if (!found) {
            System.out.println("No upper case letter");
        }
    }

    public static void printUpperCaseLetters(String str, int index, boolean found) {
        if (index == str.length()) {
            // base case: reached the end of the string
            return;
        }

        char c = str.charAt(index);
        if (Character.isUpperCase(c)) {
            // c is an upper case letter, so print it
            System.out.print(c + " ");
            found = true;
        }

        // recursive case: check the next character
        printUpperCaseLetters(str, index + 1, found);
    }
}
```

Output:

```
Enter a string: hello world

No upper case letter
```

A class Recursion has been defined to print the Fibonacci series up to a limit.

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the limit: ");
        int limit = sc.nextInt();

        int a = 0;
        int b = 1;
        printFibonacciSeries(a, b, limit);
    }

    public static void printFibonacciSeries(int a, int b, int limit) {
        if (a > limit) {
            // base case: reached the limit, so stop
            return;
        }

        System.out.print(a + " ");

        // recursive case: compute the next number in the series
        int c = a + b;
        a = b;
        b = c;
        printFibonacciSeries(a, b, limit);
    }
}
```

Output:

```
Enter the limit:

13

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144
```

Write a program in Java to accept a word. Pass it to a function magic(String str). The function checks the string for the presence of consecutive letters.

```
import java.util.Scanner;

public class MagicString
{
    public void magic(String str) {

        boolean isMagicStr = false;
        String t = str.toUpperCase();
        int len = t.length();

        for (int i = 0; i < len - 1; i++) {
            if (t.charAt(i) + 1 == t.charAt(i + 1)) {
                isMagicStr = true;
                break;
            }
        }

        if (isMagicStr)
            System.out.println("It is a magic string");
        else
            System.out.println("It is not a magic string");
    }

    public static void main(String args[]) {

        Scanner in = new Scanner(System.in);
        System.out.print("Enter word: ");
        String word = in.nextLine();

        KboatMagicString obj = new KboatMagicString();
        obj.magic(word);
    }
}
```

Output:

```
Enter word: computer

It is not a magic string

Enter word: DELHI

It is a magic string
```