Q1) Given an array of size N-1 such that it only contains distinct integers in the range of 1 to N. Find the missing element.

Input :

The first line contains integer number n ($1 \le n \le 100$).

The second line contains n integer numbers a1, a2, ..., an ($1 \le ai \le n$).

```
import java.util.Scanner;
class Main{
static int solve(int array[], int n) {
      for(int i:array)
      {
         if(i%(n+1) < n)
         array[i%(n+1)- 1] += (n+1);
      }
      for(int i=0;i<n-1;i++)
      {
         if(array[i]/(n+1) == 0)
         {
            return i+1;
         }
      }
      return n;
   }
   public static void main(String arg[])
   {
      int n;
      Scanner sc = new Scanner(System.in);
      n=sc.nextInt();
      int a[] = new int[n-1];
      for(int i=0;i<n-1;i++)
      {
          a[i] = sc.nextInt();
      }
      System.out.println(solve(a,n));
   }
   }
```

Q2) Bharat wants to buy w bananas in the shop. He has to pay k dollars for the first banana, 2k dollars for the second one and so on (in other words, he has to pay i·k dollars for the i-th banana).

He has n dollars. How many dollars does he have to borrow from his friend to buy w bananas?

Solution :

```
import java.util.Scanner;

public class Main {

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int k = sc.nextInt();
      int n = sc.nextInt();
      int w = sc.nextInt();
      int costoTotal = w * (w + 1) / 2 * k;
      int dineroFaltante = Math.max(costoTotal - n, 0);
      System.out.println(dineroFaltante);
   }
}
```

Q3) You are given sequence a1, a2, ..., an of integer numbers of length n. Your task is to find such subsequence that its sum is odd and maximum among all such subsequences. It's guaranteed that a given sequence contains a subsequence with an odd sum.

Subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements.

You should write a program which finds the sum of the best subsequence.

Input :

The first line contains integer number n ($1 \le n \le 105$).

The second line contains n integer numbers a1, a2, ..., an ($-104 \le ai \le 104$). The sequence contains at least one subsequence with odd sum.

Solution :

```
import java.util.*;
public class B1 {
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine();

        int[] nums = new int[n];
        for(int i=0; i <n; i++){
            nums[i] = sc.nextInt();
        }
        if(n == 1){
            System.out.println(nums[0]);
            return;
        }

        int minOdd = Integer.MAX_VALUE;
        int runningSum = 0;
        for(int i=0; i < n; i++){
            if(nums[i] > 0){
                runningSum += nums[i];
            }

            if(nums[i]%2 != 0){
                if(minOdd > Math.abs(nums[i])){
                    minOdd = Math.abs(nums[i]);
                }
            }
        }
        if(runningSum%2 ==0){
            runningSum -= minOdd;
        }
        System.out.println(runningSum);
    }
}
```

Q4) Given a big positive number x represented as string, find value of x % 11 or x mod 11. Output is expected as an integer.

Solution :

```
class Solution
{
   static int xmod11(String x)
{
   int n = x.length();
      int rem=0;
      for(int i=0; i<n; i++){
         rem = ( rem* 10 +(int)(x.charAt(i) - '0'))%11;
      }
      return rem;

}
}
class JavaProgram1 {
   public static void main(String args[])
   {
     public static void main(String args[]) throws IOException {
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));


      int t = Integer.parseInt(br.readLine());

      while(t > 0){

         String x = br.readLine();

         Solution ob = new Solution();

         System.out.println(ob.xmod11(x));

         t--;
      }
   }
   }
}
```

Q5) Imagine an imaginary array of length N containing balls. Given 2 arrays color and radius of length N each, where color[i] represents the color of the ith ball while radius[i] represents the radius of ith ball. If two consecutive balls have the same color and size, both are removed from the array. Geek wants to know the length of the final imaginary array.

Solution :

```
import java.io.*;
import java.util.*;
```

```
class IntArray
{
   public static int[] input(BufferedReader br, int n) throws IOException
   {
      String[] s = br.readLine().trim().split(" ");
      int[] a = new int[n];
      for(int i = 0; i < n; i++)
         a[i] = Integer.parseInt(s[i]);

      return a;
   }

   public static void print(int[] a)
   {
      for(int e : a)
         System.out.print(e + " ");
      System.out.println();
   }

   public static void print(ArrayList<Integer> a)
   {
      for(int e : a)
         System.out.print(e + " ");
      System.out.println();
   }
}

class JavaProgram{
   public static void main(String[] args) throws IOException {
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      int t;
      t = Integer.parseInt(br.readLine());
      while(t-- > 0){

         int N;
         N = Integer.parseInt(br.readLine());

         int[] color = IntArray.input(br, N);

         int[] radius = IntArray.input(br, N);

         Solution obj = new Solution();
         int res = obj.finLength(N, color, radius);

         System.out.println(res);

      }
   }
}


class Solution {
   public static int finLength(int N, int[] color, int[] radius) {
   Stack<Integer> stack = new Stack<>(); // create a stack to store ball indices
   for (int i = 0; i < N; i++) { // iterate over all balls
```

```
        if (!stack.isEmpty() && // check if the stack is not empty and
            color[i] == color[stack.peek()] && // the current ball has the same color as the top
ball in the stack and
            radius[i] == radius[stack.peek()]) { // the current ball has the same radius as the
top ball in the stack
            stack.pop(); // remove the top ball from the stack
        } else {
            stack.push(i); // add the current ball to the stack
        }
    }
    return stack.size(); // return the size of the remaining balls in the stack

    }
}
```

Q7) You are going to book a taxi. There are infinite number of points 1, 2, 3... on the X axis and your current position is cur. There are N Taxis near you, and the position of those taxis is given as an array pos. Where pos[i] denotes the position of the ith taxi. You are also given an array time. Where time[i] denotes the time taken by the ith taxi to cover 1 unit of distance. Your task is to find the minimum time to board a taxi.

Solution :

```
import java.io.*;
import java.util.*;


class IntArray
{
    public static int[] input(BufferedReader br, int n) throws IOException
    {
        String[] s = br.readLine().trim().split(" ");
        int[] a = new int[n];
        for(int i = 0; i < n; i++)
            a[i] = Integer.parseInt(s[i]);

        return a;
    }
```

```
    public static void print(int[] a)
    {
        for(int e : a)
            System.out.print(e + " ");
        System.out.println();
    }

    public static void print(ArrayList<Integer> a)
    {
        for(int e : a)
            System.out.print(e + " ");
        System.out.println();
    }
}

class JavaProgram{
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int t;
        t = Integer.parseInt(br.readLine());
        while(t-- > 0){

            int N;
            N = Integer.parseInt(br.readLine());


            int cur;
            cur = Integer.parseInt(br.readLine());


            int[] pos = IntArray.input(br, N);


            int[] time = IntArray.input(br, N);

            Solution obj = new Solution();
            int res = obj.minimumTime(N, cur, pos, time);

            System.out.println(res);

        }
    }
}



class Solution {
    public static int minimumTime(int N, int cur, int[] pos, int[] time) {
        // code here
        int k=Integer.MAX_VALUE;
        for(int i=0;i<N;i++){
            int s=(Math.abs(cur-pos[i]))*time[i];
            if(s<k){
                k=s;
            }
        }
        return k;
```

```
    }
}
```

Q8)Given an array of n elements that contains elements from 0 to n-1, with any of these numbers appearing any number of times. Find these repeating numbers in O(n) and using only constant memory space

Solution :

```
class JavaProgram2 {
    public static void main(String args[])
    {
        int numRay[] = { 0, 4, 3, 2, 7, 8, 2, 3, 1 };
        for (int i = 0; i < numRay.length; i++) {
            numRay[numRay[i] % numRay.length]
                    = numRay[numRay[i] % numRay.length]
                    + numRay.length;
        }
        System.out.println("The repeating elements are : ");
        for (int i = 0; i < numRay.length; i++) {
            if (numRay[i] >= numRay.length * 2) {
                System.out.println(i + " ");
            }
        }
    }
}
```

Q9)Write a Java program that demonstrates the use of aggregation? The program should define two classes: Author and Book. The Book class should have a reference to an Author object using aggregation. The program should allow the user to input the details of the Author and Book objects and then print out the details of the Book object, including the details of the Author object using aggregation

Solution :

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

```
        System.out.print("Enter author name: ");
        String authorName = input.nextLine();
        System.out.print("Enter author location: ");
        String authorLocation = input.nextLine();
        System.out.print("Enter author birth date: ");
        String authorBirthDate = input.nextLine();
        Author objAuthor = new Author(authorName, authorLocation, authorBirthDate);
        System.out.print("Enter book name: ");
        String bookName = input.nextLine();
        System.out.print("Enter book ID: ");
        String bookID = input.nextLine();
        Book objBook = new Book(objAuthor, bookName, bookID);
        // Book details
        System.out.println("Book Name: " + objBook.name);
        System.out.println("Book ID: " + objBook.publisherId);
        // Author details
        System.out.println("Author Name: " + objBook.author.name);
        System.out.println("Author Birth Date: " + objBook.author.birthDate);
        System.out.println("Author Location: " + objBook.author.location);

        input.close();
    }// main()
}// JavaAggregation

class Author {
    String name;
    String location;
    String birthDate;
    public Author(String name, String location, String birthDate) {
        this.name = name;
        this.location = location;
        this.birthDate = birthDate;
    } // constructor
}// Author
class Book {
    Author author;
    String name;
    String publisherId;
    public Book(Author author, String name, String publisherId) {
```

```
        this.author = author;
        this.name = name;
        this.publisherId = publisherId;
    }// constructor
}// Book
```

Q10)How can I implement encoding and decoding of a message using XOR cipher in Java?

Solution :

```
public class Decode{
public static String decode(String input, int key) {
    StringBuilder output = new StringBuilder();
    for (int i = 0; i < input.length(); i++) {
        char c = input.charAt(i);
        c = (char) (c ^ key);
        output.append(c);
    }
    return output.toString();
}
public static String encode(String input, int key) {
    StringBuilder output = new StringBuilder();
    for (int i = 0; i < input.length(); i++) {
        char c = input.charAt(i);
        c = (char) (c ^ key);
        output.append(c);
    }
    return output.toString();
}
public static void main(String ar[]){
```

```
String message = "Literals In Java";
int key = 11;
String encodedMessage = encode(message, key);
System.out.println("Encoded message: " + encodedMessage);
String decodedMessage = decode(encodedMessage, key);
System.out.println("Decoded message: " + decodedMessage);
} }
```

Q11. Given an array of size N-1 such that it only contains distinct integers in the range of 1 to N. Find the missing element.

```
import java.util.Scanner;
class Main{
    static int solve(int array[], int n) {
        // Your Code Here
        for(int i:array)
        {
            if(i%(n+1) < n)
            array[i%(n+1)- 1] += (n+1);

        }
        for(int i=0;i<n-1;i++)
        {
            //  System.out.print(array[i] + " ");
            if(array[i]/(n+1) == 0)
            {
                return i+1;
            }
        }
        return n;
    }
    public static void main(String arg[])
    {
        int n;
        Scanner sc = new Scanner(System.in);
        n=sc.nextInt();
        int a[] = new int[n-1];
        for(int i=0;i<n-1;i++)
        {
          a[i] = sc.nextInt();
        }
        System.out.println(solve(a,n));
    }
    }
```

Q12. Working with 2D arrays is quite important. Here we will do swapping of columns in a 2D array. You are given a matrix M or r rows and c columns. You need to swap the first column with the last column.

Solution:

```
import java.util.Scanner;
class Main{
    static void solve(int a[][],int r, int c){
        for(int i = 0;i<r;i++){
           int temp = a[i][0];
           a[i][0] = a[i][c-1];
           a[i][c-1] = temp;

        }
        for(int i = 0;i<r;i++){
            for(int j = 0;j<c;j++){
                System.out.print(a[i][j] + " ");
            }
            System.out.println();
        }
    }
    public static void main(String arg[])
    {
        int n,m;
        Scanner sc = new Scanner(System.in);
        n=sc.nextInt();
        m=sc.nextInt();
        int a[][] = new int[n][m];
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<m;j++)
             a[i][j] = sc.nextInt();
        }
        solve(a,n,m);
    }
    }
```

Q13. A new delivery of clothing arrived today to the clothing store. This delivery consists of a ties, b

scarves, c
vests and d
jackets.

The store does not sell single clothing items — instead, it sells suits of two types:

a suit of the first type consists of one tie and one jacket;
a suit of the second type consists of one scarf, one vest and one jacket.

Each suit of the first type costs e
coins, and each suit of the second type costs f
coins.

Calculate the maximum possible cost of a set of suits that can be composed from the delivered clothing items. Note that one item cannot be used in more than one suit (though some items may be left unused).

SOLUTION:

```
import java.util.Scanner;
public class q1
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        long a = sc.nextInt();
        long b = sc.nextInt();
        long c = sc.nextInt();
        long d = sc.nextInt();
        long e = sc.nextInt();
        long f = sc.nextInt();
        long x = Math.min(b,c);
        long ans=0;
        if(e>f)
        {
            if(a>d)
            {
             ans += (d)*e;
             d=0;
            }
            else
            {
                ans += a*e;
                d=d-a;
                ans+= Math.min(x,d)*f;
            }
        }
        else
        {
```

```
            if(x>d)
            {
                ans += d*f;
                d=0;
            }
            else
            {
                ans +=x*f;
                d=d-x;
                ans += Math.min(a,d)*e;
            }
        }
        System.out.println(ans);
    }
}
```

Q13)Given an array of integers, find the length of the longest sub-sequence such that elements in the subsequence are consecutive integers, the consecutive numbers can be in any order.

Input: arr[] = {1, 9, 3, 10, 4, 20, 2}

Output: 4

Solution :

```
class JavaProgram3 {
    static int findLongestConseqSubseq(int arr[], int n)
    {
        // Sort the array
        Arrays.sort(arr);
        int ans = 0, count = 0;
        ArrayList<Integer> v = new ArrayList<Integer>();
        v.add(arr[0]);
        // Insert repeated elements
        // only once in the vector
        for (int i = 1; i < n; i++) {
            if (arr[i] != arr[i - 1])
                v.add(arr[i]);
```

```
        }
        // Find the maximum length
        // by traversing the array
        for (int i = 0; i < v.size(); i++) {

            // Check if the current element is
            // equal to previous element +1
            if (i > 0 && v.get(i) == v.get(i - 1) + 1)
                count++;
            else
                count = 1;
            // Update the maximum
            ans = Math.max(ans, count);
        }
        return ans;
    }



    public static void main(String[] args)
    {
        int arr[] = { 1, 9, 3, 10, 4, 20, 2 };
        int n = arr.length;

        System.out.println(
              "Length of the Longest "
                    + "contiguous subsequence is "
                    + findLongestConseqSubseq(arr, n));
    }
}
```

Q14)Write java program that demonstrates the use of DecimalFormat class to format a given number? The program should take a double value and use various format specifiers to format the number with different precision and styles, and print the formatted output to the console

Solution :

```
import java.text.DecimalFormat;
class JavaProgram15 {
    public static void main(String args[])
    {
        double num = 123.4567;
```

```
        // prints only numeric part of a floating number
        DecimalFormat ft = new DecimalFormat("####");
        System.out.println("Without fraction part: num = "
                + ft.format(num));

        // this will print it upto 2 decimal places
        ft = new DecimalFormat("#.##");
        System.out.println(
                "Formatted to Give precision: num = "
                        + ft.format(num));

        // automatically appends zero to the rightmost part
        // of decimal instead of #,we use digit 0
        ft = new DecimalFormat("#.000000");
        System.out.println(
                "appended zeroes to right: num = "
                        + ft.format(num));

        // automatically appends zero to the leftmost of
        // decimal number instead of #,we use digit 0
        ft = new DecimalFormat("00000.00");
        System.out.println(
                "formatting Numeric part : num = "
                        + ft.format(num));

        // formatting money in dollars
        double income = 23456.789;
        ft = new DecimalFormat("$###,###.##");
        System.out.println("your Formatted Dream Income : "
                + ft.format(income));
    }
}
```

Q15)Write Program to simulates interaction between a microcontroller and a communication interface using bitwise operators. The microcontroller has eight output pins, and the communication interface sends binary data that the microcontroller interprets to set the state of the pins.

Solution :

```
public class Main {
```

```
public static void main(String[] args) {
    // Initialize the pins
    Pin pin0 = new Pin(0);
    Pin pin1 = new Pin(1);
    Pin pin2 = new Pin(2);
    Pin pin3 = new Pin(3);
    Pin pin4 = new Pin(4);
    Pin pin5 = new Pin(5);
    Pin pin6 = new Pin(6);
    Pin pin7 = new Pin(7);

    // Initialize the communication interface
    CommunicationInterface comm = new CommunicationInterface();
    // Main loop
    while (true) {
        // Wait for incoming data
        byte[] data = comm.receive();
        // Check if data was received
        if (data != null && data.length == 1) {
            // Extract the binary value from the received data
            int value = data[0] & 0xFF;
            // Set the state of each pin based on the binary value
            if ((value & 0x01) == 0x01) {
                pin0.setHigh();
            } else {
                pin0.setLow();
            }
            if ((value & 0x02) == 0x02) {
                pin1.setHigh();
            } else {
                pin1.setLow();
            }
            if ((value & 0x04) == 0x04) {
                pin2.setHigh();
            } else {
                pin2.setLow();
            }
            if ((value & 0x08) == 0x08) {
                pin3.setHigh();
```

```
            } else {
                pin3.setLow();
            }
            if ((value & 0x10) == 0x10) {
                pin4.setHigh();
            } else {
                pin4.setLow();
            }
            if ((value & 0x20) == 0x20) {
                pin5.setHigh();
            } else {
                pin5.setLow();
            }
            if ((value & 0x40) == 0x40) {
                pin6.setHigh();
            } else {
                pin6.setLow();
            }
            if ((value & 0x80) == 0x80) {
                pin7.setHigh();
            } else {
                pin7.setLow();
            }
        }
    }
}

class Pin {
    private int pinNumber;
    public Pin(int pinNumber) {
        this.pinNumber = pinNumber;
    }

    public void setHigh() {
        // Set the pin to a high state
        System.out.println("Pin " + pinNumber + " set to HIGH");
    }
```

```
    public void setLow() {
        // Set the pin to a low state
        System.out.println("Pin " + pinNumber + " set to LOW");
    }
}
class CommunicationInterface {
    public byte[] receive() {
        // Simulate receiving data from a communication interface
        byte[] data = null;
        double rand = Math.random();
        if (rand < 0.5) {
            data = new byte[] { (byte) 0x55 };
        } else {
            data = new byte[] { (byte) 0xAA };
        }
        System.out.println("Received data: " + Arrays.toString(data));
        return data;
    }
}
```

Q16)Given the heights of N towers and a value of K, Either increase or decrease the height of every tower by K (only once) where K > 0. After modifications, the task is to minimize the difference between the heights of the longest and the shortest tower and output its difference.

Solution :

```
public class JavaProgram4 {
    public static void main(String[] args)
    {
        int[] arr = { 7, 4, 8, 8, 8, 9 };
        int k = 6;
        int ans = getMinDiff(arr, arr.length, k);
        System.out.println(ans);
    }
    // User function Template for Java
    public static int getMinDiff(int[] arr, int n, int k)
    {
        Arrays.sort(arr);
        // Maximum possible height difference
        int ans = arr[n - 1] - arr[0];
        int tempmin, tempmax;
        tempmin = arr[0];
```

```
        tempmax = arr[n - 1];
        for (int i = 1; i < n; i++) {
            // if on subtracting k we got negative then continue
            if (arr[i] - k < 0)
                continue;
            // Minimum element when we add k to whole array
            tempmin = Math.min(arr[0] + k, arr[i] - k);
            // Maximum element when we subtract k from whole array
            tempmax  = Math.max(arr[i - 1] + k, arr[n - 1] - k);
            ans = Math.min(ans, tempmax - tempmin);
        }
        return ans;
    }
}
```

Q17)Given a matrix mat[][], the task is to check whether the matrix is valid or not. A valid matrix is the matrix that satisfies the given conditions:

For every row, it must contain a single distinct character.

No two consecutive rows have a character in common.

Solution :

```
class JavaProgram14
{

    // Storing the size of the matrix
    static final int M = 5, N = 5;

    // Function that returns true if the matrix is valid
    static boolean isPerfect(String board[])
    {
        char m = 0;
        for (int i = 0; i < M; i++)
        {
            // Get the current row
            String s = board[i];
            /* Comparing first element of the row with the element of previous row */
            if (i > 0 && s.charAt(0) == m)
                return false;
            // Checking if all the characters of the
            // current row are same or not
```

```
        for (int j = 0; j < N; j++)
        {
            // Storing the first character
            if (j == 0)
                m = s.charAt(0);
                // Comparing all the elements
                // with the first element
            else
            {
                if (m != s.charAt(j))
                    return false;
            }
        }
    }
    return true;
  }
  public static void main (String[] args)
  {
    String board[] = { "88888", "44444",
          "66666", "55555",
          "88888" };

    if (isPerfect(board))
      System.out.println("Yes");
    else
      System.out.println("No");
  }
}
```

Q18)Write Java Program to Find the Determinant of a Matrix,the Determinant of a Matrix is a real number that can be defined for square matrices only i.e, the number of rows and columns of the matrices must be equal.

Solution :

```
class JavaProgram15 {
  // Dimension of input square matrix
  static final int N = 4;
  // Function to get determinant of matrix
  static int determinantOfMatrix(int mat[][], int n)
  {
```

```
int num1, num2, det = 1, index,
        total = 1; // Initialize result
// temporary array for storing row
int[] temp = new int[n + 1];
// loop for traversing the diagonal elements
for (int i = 0; i < n; i++) {
    index = i; // initialize the index
    // finding the index which has non zero value
    while (mat[index][i] == 0 && index < n) {
        index++;
    }
    if (index == n) // if there is non zero element
    {
        // the determinant of matrix as zero
        continue;
    }
    if (index != i) {
        // loop for swapping the diagonal element row
        // and index row
        for (int j = 0; j < n; j++) {
            swap(mat, index, j, i, j);
        }
        // determinant sign changes when we shift
        // rows go through determinant properties
        det = (int)(det * Math.pow(-1, index - i));
    }
    // storing the values of diagonal row elements
    for (int j = 0; j < n; j++) {
        temp[j] = mat[i][j];
    }
    // traversing every row below the diagonal
    // element
    for (int j = i + 1; j < n; j++) {
        num1 = temp[i]; // value of diagonal element
        num2 = mat[j]
              [i]; // value of next row element
        // traversing every column of row
        // and multiplying to every row
        for (int k = 0; k < n; k++) {
```

```
                // multiplying to make the diagonal
                // element and next row element equal
                mat[j][k] = (num1 * mat[j][k])
                      - (num2 * temp[k]);
            }
            total = total * num1; // Det(kA)=kDet(A);
        }
    }
    // multiplying the diagonal elements to get
    // determinant
    for (int i = 0; i < n; i++) {
        det = det * mat[i][i];
    }
    return (det / total); // Det(kA)/k=Det(A);
}

static int[][] swap(int[][] arr, int i1, int j1, int i2,
                    int j2)
{
    int temp = arr[i1][j1];
    arr[i1][j1] = arr[i2][j2];
    arr[i2][j2] = temp;
    return arr;
}
public static void main(String[] args)
{
    int mat[][] = { { 1, 0, 2, -1 },
            { 3, 0, 0, 5 },
            { 2, 1, 4, -3 },
            { 1, 0, 5, 0 } };
    // Function call
    System.out.printf(
            "Determinant of the matrix is : %d",
            determinantOfMatrix(mat, N));
}
}
```

Q19)Given a boolean 2D array, where each row is sorted.Find the row with the maximum number of 1s.

Solution :

```
class JavaProgram5 {
   static int R = 4 ;
   static int C = 4 ;
   // Function to find the index of first index
// of 1 in a boolean array arr[]
   static int first(int arr[], int low, int high)
   {
      if(high >= low)
      {
         // Get the middle index
         int mid = low + (high - low)/2;

         // Check if the element at middle index is first 1
         if ( ( mid == 0 || arr[mid-1] == 0) && arr[mid] == 1)
            return mid;
            // If the element is 0, recur for right side
         else if (arr[mid] == 0)
            return first(arr, (mid + 1), high);
            // If element is not first 1, recur for left side
         else
            return first(arr, low, (mid -1));
      }
      return -1;
   }
   // Function that returns index of row
// with maximum number of 1s.
   static int rowWithMax1s(int mat[][])
   {
      // Initialize max values
      int max_row_index = 0, max = -1;
      // Traverse for each row and count number of 1s
      // by finding the index of first 1
      int i, index;
      for (i = 0; i < R; i++)
      {
         index = first (mat[i], 0, C-1);
         if (index != -1 && C-index > max)
```

```
            {
                max = C - index;
                max_row_index = i;
            }
        }
        return max_row_index;
    }
    public static void main(String[] args) {
        int mat[][] = { {0, 0, 0, 1},
                {0, 1, 1, 1},
                {1, 1, 1, 1},
                {0, 0, 0, 0}};
        System.out.print("Index of row with maximum 1s is " + rowWithMax1s(mat));
    }
}
```

Q19)Given an array A[] of N integers such that A[0] + A[1] + A[2] + … A[N – 1] = 0. The task is to generate an array B[] such that B[i] is either ⌊A[i] / 2⌋ or ⌈A[i] / 2⌉ for all valid i and B[0] + B[1] + B[2] + … + B[N – 1] = 0.

Solution :

```
class JavaProgram13
{
    static void printArr(int arr[], int n)
    {
        for (int i = 0; i < n; i++)
            System.out.print(arr[i] + " ");
    }
    // Function to generate and print
// the required array
    static void generateArr(int arr[], int n)
    {

        // To switch the ceil and floor
        // function alternatively
        boolean flip = true;
        // For every element of the array
        for (int i = 0; i < n; i++)
        {
            // If the number is odd then print the ceil
            // or floor value after division by 2
```

```
            if ((arr[i] & 1) != 0)
            {
                // Use the ceil and floor alternatively
                if (flip ^= true)
                    System.out.print((int)(Math.ceil(arr[i] /
                            2.0)) + " ");
                else
                    System.out.print((int)(Math.floor(arr[i] /
                            2.0)) + " ");
            }
            // If arr[i] is even then it will
            // be completely divisible by 2
            else
            {
                System.out.print(arr[i] / 2 +" ");
            }
        }
    }
    public static void main(String []args)
    {
        int arr[] = { 3, -5, -7, 9, 2, -2 };
        int n = arr.length;

        generateArr(arr, n);
    }
}
```

Q20)Create a class named 'Rectangle' with two data members 'length' and 'breadth' and two methods to print the area and perimeter of the rectangle respectively. Its constructor having parameters for length and breadth is used to initialize length and breadth of the rectangle. Let class 'Square' inherit the 'Rectangle' class with its constructor having a parameter for its side (suppose s) calling the constructor of its parent class as 'super(s,s)'. Print the area and perimeter of a rectangle and 10 squares.

Solution :

```
import java.util.Scanner;
class Rectangle{
    int length;
    int breadth;
    public Rectangle(int l, int b){ //parameterized constuctor
```

```
        length = l;
        breadth = b;
    }

    public void printArea(){
        System.out.println(length*breadth); //print area
    }

    public void printPerimeter(){
        System.out.println(2*(length+breadth)); //print perimeter
    }
}

class Square extends Rectangle{
    int side;
    public Square(int s){
        super(s,s); //square has same size of sides so pass it to base Rectangle
    }
}

class JavaProgram2{
    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
        System.out.print("Enter length of rectangle: ");
        int length = input.nextInt();
        System.out.print("Enter breadth of rectangle: ");
        int breadth = input.nextInt();
        Rectangle r = new Rectangle(length, breadth);
        r.printArea();
        r.printPerimeter();
        System.out.print("Enter side of square: ");
        int side = input.nextInt();
        Square s = new Square(side);
        s.printArea();
        s.printPerimeter();

        Square[] a = new Square[10]; //array of objects
        int k = 5;
```

```
        //create 10 objects of square
        for(int i = 0;i<10;i++){
            a[i] = new Square(k);
            k++;
        }

        //print area and perimeter of 10 squares
        for(int i = 0;i<10;i++){
            a[i].printArea();
            a[i].printPerimeter();
        }
        input.close();
    }
}
```