1.check if the array is sorted in forward or backward or not at all:

Program:

```
class Main{
   public static String checkArrayOrder(int[]arr){
      if(arr == null||arr.length<2){
         return"Array is too short to handle";
      }


      boolean isAscending =true;
      boolean isDescending=true;
      for(int i=1;i<arr.length;i++){
      if(arr[i]>arr[i-1]){
         isDescending =false;
      }
       if(arr[i]<arr[i-1]){
         isAscending =false;
      }
      }
      if(isAscending){
         return "elements are in ascending order";
      }
      if(isDescending){
         return "elements are in descending order";
      }
      return "Array not sorted";
   }
      public static void main (String[]args){
         int [] arr = {6,5,4,3,2,1};
         int []arr2 = {1,2,3,4,5,6};
         int [] arr3 ={1,3,2,4,1,3};
         System.out.println(checkArrayOrder(arr));
```

```
        System.out.println(checkArrayOrder(arr2));
        System.out.println(checkArrayOrder(arr3));      }



    }
```

Output:

```
elements are in descending order
elements are in ascending order
Array not sorted
```

2. Find second maximum element if none return -1;

Program:

```
class Main {
  public static int findSecondMax(int[] arr) {
    if (arr.length < 2) return -1; // Not enough elements

    int max = Integer.MIN_VALUE;    // Initialize maximum
    int secondMax = Integer.MIN_VALUE; // Initialize second maximum

    for (int num : arr) {
      if (num > max) {
        secondMax = max; // Update second maximum
        max = num;      // Update maximum
      } else if (num > secondMax && num < max) {
        secondMax = num; // Update second maximum if it's smaller than max but larger than
current secondMax
      }
    }

    return secondMax == Integer.MIN_VALUE ? -1 : secondMax; // If no second max, return -1
  }
```

```
    public static void main(String[] args) {
        int[] arr = {3, 5, 7, 2, 8};
        int secondMax = findSecondMax(arr);
        System.out.println("The second maximum element in the array is: " + secondMax);
    }
}
```

Output: The second maximum element in the array is: 7

3. create duplicate of an array

Program:

```
public class DuplicateArray {
    public static void main(String[] args) {
        int[] original = {1, 2, 3, 4, 5};  // Original array
        int[] duplicate = new int[original.length];  // Create a new array of the same size

        for (int i = 0; i < original.length; i++) {
            duplicate[i] = original[i];  // Copy each element
        }

        System.out.print("Duplicate array: ");
        for (int num : duplicate) {
            System.out.print(num + " ");
        }
    }
}
```

>…>>>>> Alternate Method:

```
public class DuplicateArray {
    public static void main(String[] args) {
        int[] original = {1, 2, 3, 4, 5};  // Original array
        int[] duplicate = new int[original.length];  // Create a new array of the same size

        System.arraycopy(original, 0, duplicate, 0, original.length);

        System.out.print("Duplicate array: ");
        for (int num : duplicate) {
            System.out.print(num + " ");
        }
    }
}
```

Output:

Given input

int[] original = {1, 2, 3, 4, 5};

formed output :

```
Duplicate array: 1 2 3 4 5
```

4. Program to demonstrate unique and duplicate elements

Program:

```
class Main{
    public static void UniqueandDuplicateElements(int[]arr){
        int [] frequency = new int[101];
        int uniquecount= 0;
        int duplicatecount = 0;
        for(int num:arr){
            frequency[num]++;
        }
        for(int i =0; i<frequency.length;i++){
            if(frequency[i]==1){
                uniquecount++;
            }
            else if(frequency[i]>1){
                duplicatecount++;
            }
            }
            System.out.println(uniquecount);
            System.out.println(duplicatecount);
        }
        public static void main(String[]args){
            int [] arr ={1,2,3,2,3,4,1,5,6,2,3,4};
             UniqueandDuplicateElements(arr);
        }
    }
```

Output:

```
2
4
```

5.insert an element at xth position shifting right:

```
import java.util.Arrays;

class Main {
    public static int[] insertAtPosition(int[] arr, int element, int position) {
        int[] newArr = new int[arr.length + 1];
        for (int i = 0; i < position; i++) newArr[i] = arr[i];
        newArr[position] = element;
        for (int i = position; i < arr.length; i++) newArr[i + 1] = arr[i];
        return newArr;
    }

    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        int element = 9;
        int position = 2;

        int[] newArr = insertAtPosition(arr, element, position);

        System.out.println(Arrays.toString(newArr));

    }

}
```

Output: 1,2,9,3,4,5

6. Delete an element at the Xth position, shifting left.

```
class Main {

    public static int[] deleteAtPosition(int[] arr, int position) {

        for (int i = position; i < arr.length - 1; i++) {

            arr[i] = arr[i + 1]; // Shift elements to the left

        }

        arr[arr.length - 1] = 0; // Replace the last element with 0 (placeholder)

        return arr;

    }


    public static void main(String[] args) {

        int[] arr = {1, 2, 3, 4, 5};

        int position = 2; // Delete the element at index 2 (value 3)

        int[] result = deleteAtPosition(arr, position);

         System.out.println(Arrays.toString(result));    }

}
```

Output: 12450

>>>>>Alternate method :

>>>>>USING ARRAYLIST:

```
import java.util.ArrayList;

import java.util.Arrays;


public class DeleteElement {

    public static void main(String[] args) {

        ArrayList<Integer> arr = new ArrayList<>(); // Declare the ArrayList
        arr.addAll(Arrays.asList(1, 2, 3, 4, 5));   // Populate it on the next line


        int position = 2; // Delete the element at index 2

        arr.remove(position);

        System.out.println(arr);

// Output: [1, 2, 4, 5]

    }

}
```

Strings :

1.print ASCII values of each character in the given string:

Program:

```
class Main{

  public static void main(String[]args){

    String s = "Hello";

    for(int i=0;i<s.length();i++){

    System.out.println("ASCII value of "+s.charAt(i)+"is"+(int)s.charAt(i));

  }

  }

}
```

Output:

```
ASCII value of His72
ASCII value of eis101
ASCII value of lis108
ASCII value of lis108
ASCII value of ois111
```

2.

Count letters, numbers, and special characters in a string.

```
class Main{
  public static void main(String []args){
    String s = "sai7@";
    int letters = 0;
    int numbers= 0;
    int specialchars = 0;
    for(int i=0;i<s.length();i++){
      int ch = s.charAt(i);
      if(ch>='A'& ch<='Z' || ch>='a'& ch<='z'){
        letters++;
      }
    else if(ch>='0' && ch<='9'){
      numbers++;
    }
    else{
      specialchars++;
    }
    }
    System.out.println( "number of letters in the string are: "+letters);
    System.out.println("total numbers in string are:"+numbers);
    System.out.println("total number of special characters in string are:"+specialchars);
  }
}
```

Output:

```
number of letters in the string are: 3
total numbers in string are:1
total number of special characters in string are:1
```

Another approach :

```
class Main{
    public static void main(String []args){
        String s = "bhaskar7@";
        int vowels = 0;
        int specialChars = 0;
        int numbers = 0;
        char [] ch = {'a','e','i','o','u'};
        for(int i =0;i<s.length();i++){
           char l = s.charAt(i);
            if((String.valueOf(ch)).contains(String.valueOf(l))){
                vowels+=1;
            }
            else if(Character.isDigit(l)){
            numbers+=1;
        }
        else if(!Character.isLetter(l)){
             specialChars+=1;
        }
    }
    System.out.println("vowels in the given string are: "+vowels);
    System.out.println("numbers in the given string are: "+numbers);
    System.out.println("specialChars in the given string are:
"+specialChars);
```

```
Output:
vowels in the given string are: 2
numbers in the given string are: 1
specialChars in the given string are: 1

=== Code Execution Successful ===
```

3. program to find the difference between vowels and consonants in the string

Program:

```
class Main {
    public static void main(String[] args) {
        String s = "bhaskar7@";
        int vowels = 0;
        int consonants = 0; // Declare consonants
        char[] ch = {'a', 'e', 'i', 'o', 'u'};

        for (int i = 0; i < s.length(); i++) {
            char l = Character.toLowerCase(s.charAt(i)); // Convert to
lowercase for uniformity
            if (String.valueOf(ch).contains(String.valueOf(l))) {
                vowels += 1; // Count vowels
            } else if (Character.isLetter(l)) { // Check for alphabetic
characters that are not vowels
                consonants += 1; // Count consonants
            }
        }

        int difference = vowels - consonants; // Calculate the difference
        System.out.println("Difference between vowels and consonants: " +
difference);
    }
}
```

```
Output:
Difference is: -7
```

4. program to display sum of numbers in the string

```
class Main{

  public static void main(String[]args){
        String s = "sairam123";
        int sum = 0;
        for(int i = 0;i<s.length();i++){
                char  ch = s.charAt(i);

            if(Character.isDigit(ch))
            sum+= Character.getNumericValue(ch);
        }
        System.out.println("sum: "+sum);



    }
}
```

Output:

```
Sum:6
```

5. program to convert lower case to upper case :

```
class Main{
    public static void main(String[]args){
        String s = "bhaskar";
       String  uppercaseString = "";
        for(int i =0;i<s.length();i++){
        char ch = Character.toUpperCase(s.charAt(i));
        uppercaseString+= ch;
    }
    System.out.println("after converting the string into Uppercase :"+uppercaseString);
    }
}
```

Output:BHASKAR

6. Convert uppercase to lowercase and vice versa in a string.

```
class Main{
    public static void main(String[]args){
        String s = "BhAsKaR123";
        String result ="";
        for(int i =0;i<s.length();i++){
          char ch = s.charAt(i);
            if(Character.isUpperCase(ch))
            result+=Character.toLowerCase(ch);
            else if(Character.isLowerCase(ch))
            result+=Character.toUpperCase(ch);
            else{
                result+=ch;
            }
        }
        System.out.println(result);
    }
}
```

Output:bHaSkAr123

7. Remove leading, trailing, and extra spaces in a string.

Program:

```
class Main{
    public static void main(String[] args){
        String s = "hello  bhaskar  sai  ram";
         s =s.trim();
         s =s.replaceAll("\\s+"," ");
         System.out.println("String after removing  extra spaces:\""+s+"\"");
    }
}
```

Output:

```
String after removing  extra spaces:"hello bhaskar sai ram"
```

8. count number of words in the given string

Program:

```
class Main{
    public static void main(String[]args){
        String s = "bhaskar sai ram";
        s.trim();
        String []word = s.split("\\s+");
        int wordCount = word.length;
        System.out.println("number of words in the given string are: "+wordCount);
    }
}
```

Output:

```
number of words in the given string are: 3
```

9.count number of letters in the given string

Program:

```
class Main{
    public static void main(String[]args){
        String s = "bhaskar";
        int count = 0;
        for(int i = 0;i<s.length();i++){
            if(Character.isLetter(s.charAt(i))){
                count+=1;
            }
        }
        System.out.println("number of letters in the given string:"+count);
    }
}
```

Output:

```
number of letters in the given string:7
```

10 . print the min and max frequency element

```
import java.util.HashMap;
class Main{
    public static void main(String[]args){
      HashMap<Integer,Integer>map = new HashMap<>();
      int [] arr = {1,2,3,2,3,2,2,2,2};
      for(int num:arr){
        if(map.containsKey(num))
        map.put(num,map.get(num)+1);
        else{
          map.put(num,1);
        }
      }
       int maxFreq = 0;
        int minFreq = Integer.MAX_VALUE;
      for(int num:map.keySet()){


        maxFreq = Math.max(maxFreq,map.get(num));
        minFreq = Math.min(minFreq,map.get(num));
      }
      System.out.println("max number is : "+maxFreq);
      System.out.println("min number is : "+minFreq);







    }
  }
```

Output:

```
max number is : 6
min number is :1
```

7. Check if there are two or three consecutive identical characters in a string.

Program:

```
class Main{
   public String CheckConsecutive(String s){
      if(s == null || s.length()==0){
         return "string cant be defined";
      }
      for(int i = 0;i<s.length();i++){
         if(i<s.length()-1&& s.charAt(i)==s.charAt(i+1))
         return "two consecutive characters are found: "+s.charAt(i);
      }
      return "no two consecutive characters are found ";
   }
   public static void main(String[]args){
      Main obj = new Main();
      String input = "abaab";
      System.out.println(obj.CheckConsecutive(input));
      String input2 = "acdba";
      System.out.println(obj.CheckConsecutive(input2));
   }
}
```

Output-

```
two consecutive characters are found: a
no two consecutive characters are found
```

8. leetcode number -28

>>>>Find the first occurance of the string

```
>>>>class Solution {
   public int strStr(String haystack, String needle) {
      // If needle is empty, return 0 as per the problem specification.
      if (needle.isEmpty()) {
         return 0;
```

```
        }

        // Loop through haystack and check for the first occurrence of needle
        for (int i = 0; i <= haystack.length() - needle.length(); i++) {
            // Compare the substring of haystack starting from index i with needle
            if (haystack.substring(i, i + needle.length()).equals(needle)) {
                return i; // Return the index of the first occurrence
            }
        }

        // If no occurrence is found, return -1
        return -1;
    }

    public static void main(String[] args) {
        Solution obj = new Solution();
        String haystack = "sadbutsad";
        String needle = "sad";

        // Test the strStr method
        System.out.println(obj.strStr(haystack, needle)); // Output should be 0, as "sad" starts at index 0
    }
}
```

Ouput

```
0
```

9. Find the first and last index of occurrence for each character in a string.

Program:

```
class Main {
    public String Solution(String s) {
        if (s == null || s.isEmpty())
            return "string cant be defined";
        for (int i = 0; i < s.length() - 1; i++) {
            for (int j = i + 1; j < s.length(); j++) {
                if (s.charAt(i) == s.charAt(j))
                    return "index of the repeated character is: " + j; // Corrected this line
            }
        }

        return "no character is repeated";
    }

    public static void main(String[] args) {
        Main obj = new Main();
        String input = "bhaskar";
        System.out.println(obj.Solution(input));
    }
}
```

Output:

```
index of the repeated character is: 4
```

10.>> Check if a string contains all letters from 'a' to 'z'.

Program:

```
import java.util.*;
class Main{
    public String Characters(String s){
        if(s.length()==0)
        return "string cant be determined";
        HashSet<Character>set = new HashSet<>();
        for(int i=0;i<s.length();i++){
            char ch = s.charAt(i);
            if(ch>='a'&& ch<='z')
            set.add(ch);
        }
        if(set.size()==26)
        return"String contains all the characters from a to z";
        else{
        return "string doesnt contains all the characters from a to z";
        }
    }
    public static void main(String []args){
        Main obj = new Main();
        String input = "bhaskar";
        String input2 = "sairam";
        System.out.println(obj.Characters(input));
        System.out.println(obj.Characters(input2));
    }
}
```

Output:

```
string doesnt contains all the characters from a to z
string doesnt contains all the characters from a to z
```

>>>>insert an element at specific position in the string:

```
class Main{
    public static String Insert(String str,char ele,int position){
        if(position<0||position>str.length())
        return"Invalid position";
        return str.substring(0,position) + ele +str.substring(position);
    }
    public static void main(String[]args){
        String str = "bhaskar";
        char ele = 's';
        int position = 4;
        String st = Insert(str,ele,position);
        System.out.println(st);
    }
}
```

Ouput

```
Bhasskar
```

>>>>>>> insert element at specific predefined position (Additional program)

```
class Main {
    public static String Insert(String str, char ele, int position, int position2, int k) {
        if (position < 0 || position > str.length() || position2 < 0 || position2 > str.length() || k < 0 || k > str.length()) {
            return "Invalid position";
        }

        // Insert at the first position
        String result = str.substring(0, position) + ele + str.substring(position);

        // Insert at the second position
        result = result.substring(0, position2 + 1) + ele + result.substring(position2 + 1);

        // Insert at the kth position
        result = result.substring(0, k) + ele + result.substring(k);

        return result;
    }

    public static void main(String[] args) {
        String str = "bhaskar";
        char ele = 's';
        int position = 0; // Insert at the beginning
        int position2 = 6; // Insert at position 6
        int k = 3; // Insert at position 3
        String st = Insert(str, ele, position, position2, k);
        System.out.println(st);
    }
}
```

Output

```
Sbhsaskasr
```

>>> Insert a character at the first, last, and Kth position in a string.

```
class Main {
    public static String insertAtPositions(String str, char element, int k) {
        if (k < 0 || k >= str.length()) {
            return "Invalid Kth position.";
        }

        // Insert at the first position
        String result = element + str;

        // Insert at the last position
        result = result + element;

        // Insert at Kth position
        result = result.substring(0, k + 1) + element + result.substring(k + 1);

        return result;
    }

    public static void main(String[] args) {
        String str = "bhaskar";
        char element = 's';
        int k = 3; // Position where 's' will be inserted

        String updatedString = insertAtPositions(str, element, k);
        System.out.println(updatedString); // Output: "sbhsaskars"
    }
}
```

Output;

```
"sbhsaskars"
```

Remove the first, last, and Kth character from a string.

Program:

```
class Main {
    public static String removeChars(String str, int k) {
        if (str == null || str.length() < 3) {
            return "String too short to remove first, last, and Kth characters!";
        }
        if (k < 1 || k >= str.length() - 1) {
            return "Invalid position for K!";
        }


        String result = str.substring(1, k) + str.substring(k + 1, str.length() - 1);
        return result;
    }

    public static void main(String[] args) {
        String str = "bhaskar";
        int k = 2;
        System.out.println(removeChars(str, k));
    }
}
```

Output – hska

----→Find a specific substring within a string.

```
class Main{
   public static String substring(String str ,int k){
      if(k<0 || k>str.length())
      return "substring is not possible";
     String  result = str.substring(0,k+1);
     return result;
   }
   public static void main(String[]args){
      String str = "bhaskar";
      int k = 2;
     String result = substring(str,k);
      System.out.println(result);


   }
}
```

Output – bhas

----------------------------→>>>>>>>>>MATRIX <<<<<<<<<<<<---

1.Print a matrix row-wise and column-wise.

Program

```
class Main {
    public static void main(String[] args) {
        int[][] matrix = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        // Print matrix row-wise
        System.out.println("Matrix printed row-wise:");
        for (int i = 0; i < matrix.length; i++) { // Iterate over rows
            for (int j = 0; j < matrix[i].length; j++) { // Iterate over columns
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println(); // Move to the next row
        }

        // Print matrix column-wise
        System.out.println("\nMatrix printed column-wise:");
        for (int j = 0; j < matrix[0].length; j++) { // Iterate over columns
            for (int i = 0; i < matrix.length; i++) { // Iterate over rows
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println(); // Move to the next column
        }
    }
}Output : Matrix printed row-wise:
1 2 3
4 5 6
7 8 9
Matrix printed column-wise:
1 4 7
2 5 8
3 6 9
```

2. print the sum of the elements in the matrix

Program

```
class Main{
    public static void main(String[]args){
        int [] []arr = {{1,2,3},{4,5,6},{7,8,9}};
        int sum = 0;
        for(int i = 0;i<arr.length; i++){
            for(int j=0;j<arr[i].length;j++){
                sum+=arr[i][j];
            }
        }
         System.out.print(sum);

    }
}
```

Output

```
45
```

Find the maximum and minimum values in each row of a matrix.

Program

```
class Main {
    public static void main(String[] args) {
        int[][] matrix = {
            {3, 8, 1},
            {4, 7, 9},
            {2, 5, 6}
        };

        // Find and print max and min values for each row
        for (int i = 0; i < matrix.length; i++) {
            int max = matrix[i][0]; // Initialize max to the first element of the row
            int min = matrix[i][0]; // Initialize min to the first element of the row

            for (int j = 1; j < matrix[i].length; j++) {
                if (matrix[i][j] > max) {
                    max = matrix[i][j]; // Update max if current element is greater
                }
                if (matrix[i][j] < min) {
                    min = matrix[i][j]; // Update min if current element is smaller
                }
            }

            // Print the maximum and minimum values for the current row
            System.out.println("Row " + (i + 1) + " -> Max: " + max + ", Min: " + min);
        }
    }
}
```

Output

```
Row 1 -> Max: 8, Min: 1
Row 2 -> Max: 9, Min: 4
Row 3 -> Max: 6, Min: 2
```

Add and subtract two matrices.

```
class Main {
    public static void main(String[] args) {
        // Define two matrices
        int[][] matrixA = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        int[][] matrixB = {
            {9, 8, 7},
            {6, 5, 4},
            {3, 2, 1}
        };

        // Matrix addition
        int[][] sum = new int[matrixA.length][matrixA[0].length];
        for (int i = 0; i < matrixA.length; i++) {
```

```
        for (int j = 0; j < matrixA[0].length; j++) {
            sum[i][j] = matrixA[i][j] + matrixB[i][j];
        }
    }

    // Matrix subtraction
    int[][] difference = new int[matrixA.length][matrixA[0].length];
    for (int i = 0; i < matrixA.length; i++) {
        for (int j = 0; j < matrixA[0].length; j++) {
            difference[i][j] = matrixA[i][j] - matrixB[i][j];
        }
    }

    // Print results
    System.out.println("Matrix A:");
    printMatrix(matrixA);

    System.out.println("\nMatrix B:");
    printMatrix(matrixB);

    System.out.println("\nSum of Matrix A and B:");
    printMatrix(sum);

    System.out.println("\nDifference of Matrix A and B:");
    printMatrix(difference);
}

// Helper method to print a matrix
public static void printMatrix(int[][] matrix) {
    for (int[] row : matrix) {
        for (int element : row) {
            System.out.print(element + " ");
            }
            System.out.println();
        }
    }
}
```

Output

```
Sum of a and b
10 10 10
10 10 10
10 10 10

Difference of a and b
-8 -6 -4
-2  0  2
 4  6  8
```

>>>Calculate the sum of each row and each column in a matrix.

Program

```
class Main{
    public static void main(String[]args){
        int[][]arr = {
            {1,2,3},{4,5,6},{7,8,9}
        };
        for(int i=0;i<arr.length;i++){
            int rowsum = 0;
            for(int j = 0;j<arr[i].length;j++){
                rowsum+=arr[i][j];
            }
        System.out.println("row"+(i+1)+ " :"+rowsum);
        }
        for(int j = 0;j<arr[0].length;j++){
                int coloumnsum = 0;
               for(int i =0;i<arr.length;i++){
                   coloumnsum+=arr[i][j];
               }
                   System.out.println("coloumn"+(j+1)+ ”:”+coloumnsum);
                     }
    }
}
```

Output :

```
row1 :6
row2 :15
row3 :24
coloumn1 :12
coloumn2 :15
coloumn3 :18
```

>>>>Find the maximum and minimum values in each column of a matrix.

Program

```
class Main{
    public static void main(String[]args){
        int [][]arr = {
            {1,2,3},{4,5,6},{7,8,9}
        };
        for(int j= 0;j<arr[0].length;j++){
            int max = arr[0][j];
            int min = arr[0][j];

        for(int i= 0;i<arr.length;i++){
            if(arr[i][j]>max)
            max = arr[i][j];
          if (arr[i][j]<min)
          min = arr[i][j];
        }
         System.out.println("coloumn" +(j+1)+ ": MAX"+max+ ": MIN"+min);
        }
    }
    }
```

Output

```
coloumn1: MAX7: MIN1
coloumn2: MAX8: MIN2
coloumn3: MAX9: MIN3

=== Code Execution Successful ===
```

>>>>Print the upper triangle and lower triangle of a matrix.

Program

```
class Main{
    public static void main(String []args){
        int [][]arr = {
            {1,2,3},{4,5,6},{7,8,9}
        };
         System.out.println("Upper Triangle:");
        for(int i =0;i<arr.length;i++){
            for(int j =0;j<arr[i].length;j++){
                if(j>=i)
                System.out.print(arr[i][j]+" ");
                else{
                    System.out.print("  ");
                }
            }
            System.out.println();
        }
        System.out.println("\nLower Triangle:");
    for(int i = 0;i<arr.length;i++){
      for(int j=0;j<arr[i].length;j++){
        if(i>=j)
        System.out.print(arr[i][j]+" ");
        else{
          System.out.print("  ");
        }
      }
      System.out.println();
    }
}
}
```

Ouput

```
Upper Triangle:

1 2 3

  5 6

    9


Lower Triangle:

1

4 5

7 8 9
```

>>>>Print the left and right diagonals of a matrix.

Program

```
class Main{
  public static void main(String[]args){
    int[][]arr ={
      {1,2,3},{4,5,6},{7,8,9}
    };
    System.out.println("left diagonal");
```

```
        for(int i =0;i<arr.length;i++){
           for(int j=0;j<arr[i].length;j++){
           if(i==j){
              System.out.print(arr[i][j]+" ");
           }
           else{
              System.out.print("  ");
           }
           }
           System.out.println();
        }
        System.out.println("\nRight diagonal");
        for(int i =0;i<arr.length;i++){
           for(int j=0;j<arr[i].length;j++){
          if((i+j) == arr.length-1)
          System.out.print(arr[i][j]+" ");
     }
        }


}
}
```

Output

```
left diagnoal
1
  5
    9

Right diagnoal
3 5 7
```

>>Sort the matrix row-wise and column-wise.

Program

```java
import java.util.Arrays;

class Main {
    public static void main(String[] args) {
        int[][] matrix = {
            {5, 4, 7},
            {1, 3, 8},
            {2, 9, 6}
        };

        System.out.println("Original Matrix:");
        printMatrix(matrix);

        // Row-wise sort
        for (int[] row : matrix) {
            Arrays.sort(row);
        }

        // Column-wise sort
        for (int j = 0; j < matrix[0].length; j++) {
            int[] column = new int[matrix.length];
            for (int i = 0; i < matrix.length; i++) {
                column[i] = matrix[i][j];
            }
            Arrays.sort(column);
            for (int i = 0; i < matrix.length; i++) {
                matrix[i][j] = column[i];
            }
        }
        System.out.println("\nSorted Matrix:");
        printMatrix(matrix);
    }

    private static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            System.out.println(Arrays.toString(row));
        }
    }
}
```

Output

```
Original Matrix:
[5, 4, 7]
[1, 3, 8]
[2, 9, 6]
Sorted Matrix:
[1, 3, 7]
[2, 5, 8]
[4, 6, 9]
```

>>>Print the matrix in a zig-zag pattern.

Program

```
class Main{
    public static void main(String [] args){
        int[][] arr = {
            {1,2,3},{4,5,6},{7,8,9}
        };
        System.out.println(" zig zag pattern ");
        for(int i=0;i<arr.length;i++){
            if(i%2==0){
                for(int j=0;j<arr[i].length;j++){
                System.out.print(arr[i][j]+" ");
                }
        }
        else{
            for(int j =arr.length-1;j>=0;j--){
                System.out.print(arr[i][j]+" ");
            }
        }
        }
    }
}
```

```
    Output
zig zag pattern
1 2 3 6 5 4 7 8 9
```

>>>> Check if a matrix is symmetric.class Main {

```
    public static void main(String[] args) {
        int[][] matrix = {
            {1, 2, 3},
            {2, 4, 5},
            {3, 5, 6}
        };

        if (isSymmetric(matrix)) {
            System.out.println("The matrix is symmetric.");
        } else {
            System.out.println("The matrix is not symmetric.");
        }
    }

    public static boolean isSymmetric(int[][] matrix) {
        // Check if the matrix is square
        int rows = matrix.length;
        for (int[] row : matrix) {
            if (row.length != rows) {
                return false;
            }
        }

        // Check symmetry
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < i; j++) { // Compare only elements below the diagonal
                if (matrix[i][j] != matrix[j][i]) {
                    return false;
                }
            }
        }
        return true;
    }
}
```

Output

```
zig zag pattern
1 2 3 6 5 4 7 8 9
```

>>>>check if a matrix is an identity matrix

Program

```
class Main{
    public static boolean isIdentitymatrix(int[][]arr){
        int rows = arr.length;
        for(int[]row:arr){
            if(row.length!=rows)
            return false;
        }
        for(int i =0;i<arr.length;i++){
```

```
      for(int j=0;j<arr[i].length;j++){
        if(i==j&&arr[i][j]!=1)
        return false;
        if(i!=j&&arr[i][j]!=0)
        return false;
      }
    }
    return true;
  }

    public static void main(String[]args){
     int[][]arr = {
       {1,0,0},{0,1,0},{0,0,1}
     };
     if(isIdentitymatrix(arr))
     System.out.println("identity matrix");
     else{
       System.out.println("Not an identity matrix");
     }
    }
}
```

Output

```
Identity matrix
```

>>>Check if a matrix is sparse (mostly zeroes).

Program

```
class Main{
  public static boolean Sparsematrix(int[][]arr){
    int zerocount = 0;
    for(int i=0;i<arr.length;i++){
      for(int j=0;j<arr[i].length;j++){
```

```
                if(arr[i][j]==0)
                zerocount++;

            }
        }
        return zerocount>(arr.length*arr[0].length/2);

        }
        public static void main(String[]args){
            int[][]arr ={
                {1,0,0},{0,0,0},{0,1,0}
            };

            if(Sparsematrix(arr))
            System.out.println("it is a sparse matrix");
            else{
                System.out.println("not a sparse matrix");
            }
        }

    }
```

Output

```
it is a sparse matrix
```

>>>>>Find the inverse of a matrix.

Program

```
class Main {
    public static void main(String[] args) {
        double[][] matrix = {
            {4, 7},
            {2, 6}
        };
```

```
// Calculate determinant
double determinant = (matrix[0][0] * matrix[1][1]) - (matrix[0][1] * matrix[1][0]);

if (determinant == 0) {
    System.out.println("Matrix is singular, no inverse exists.");
    return;
}

// Calculate adjugate
double[][] adjugate = {
    {matrix[1][1], -matrix[0][1]},
    {-matrix[1][0], matrix[0][0]}
};

// Calculate inverse
double[][] inverse = new double[2][2];
for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 2; j++) {
        inverse[i][j] = adjugate[i][j] / determinant;
    }
}

// Print inverse matrix
System.out.println("Inverse of the matrix:");
for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 2; j++) {
        System.out.print(inverse[i][j] + " ");
    }
    System.out.println();
}
```

```
    }

}
```

Output

```
Inverse of the matrix:

0.6 -0.7

-0.2 0.4
```

>>>Check if a number is an Armstrong number.

Program

```
class Main{

    public static Boolean isBoolean(int num){

        int original = num;

        int sum =0;

        int digits = String.valueOf(num).length();

        while(num>0){

            int digit = num%10;

            sum+= Math.pow(digit,digits);

            num/=10;

        }

        return sum == original;

    }

    public static void main(String[]args){

        int num = 153;

        if(isBoolean(num))

        System.out.println("it is a armstrong number");

        else{

            System.out.println("it is not an armstrong number");

        }

    }
}
```

Output

```
it is a armstrong number
```

>>>>>Count the total occurrences of the digit '1' in all positive integers less than or equal to a given integer n.

Program

```
class Main{
  public static int count(int num){
    int digitcount=0;
    while(num>0){
      int digit =num%10;
      if(digit==1){
        digitcount+=1;
      }
      num/=10;
  }
     return digitcount;


}
public static void main(String[]args){
  int num =20;
  int totalcount =0;
  for(int i=0;i<num;i++){
    totalcount+=count(i);
  }
    System.out.println("total occurance of digit i :"  +totalcount);


}


}
```

Output’

```
total occurance of digit i :12
```

>>>>Calculate the LCM and GCD of two numbers.

Program

```
import java.util.Scanner;

class Main {
   // Method to calculate GCD using Euclid's algorithm
   public static int gcd(int a, int b) {
      while (b != 0) {
         int temp = b;
         b = a % b; // Corrected the variable declaration
         a = temp;
      }
      return a;
   }

   // Method to calculate LCM using the relationship LCM * GCD = a * b
   public static int lcm(int a, int b) {
      return (a * b) / gcd(a, b);
   }

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.println("Enter the first number:");
      int num1 = sc.nextInt(); // Corrected syntax for variable names

      System.out.println("Enter the second number:");
      int num2 = sc.nextInt(); // Corrected syntax for variable names
```

```
        int gcdValue = gcd(num1, num2); // Fixed spacing and variable names
        int lcmValue = lcm(num1, num2); // Removed redundant declaration

        System.out.println("GCD of " + num1 + " and " + num2 + " is: " + gcdValue); // Fixed concatenation syntax
        System.out.println("LCM of " + num1 + " and " + num2 + " is: " + lcmValue); // Fixed concatenation syntax

        sc.close(); // Closing the scanner to avoid resource leak
    }
}
```

Output

```
Enter the first number:
12
Enter the second number:
14
GCD of 12 and 14 is: 2
LCM of 12 and 14 is: 84
```

>>>>Check if two numbers are co-prime.

Program

```
import java.util.Scanner;
class Main{
    public static int gcd(int a,int b){
        while(b!=0){
            int temp =b;
            b =a%b;
            a =temp;
        }
        return a;
    }
    public static void main(String[]args){
        Scanner sc  = new Scanner(System.in);
        System.out.println("Enter the num 1 : ");
        int num1 = sc.nextInt();
        System.out.println("Enter the num2: ");
        int num2 = sc.nextInt();
        if(gcd(num1,num2)==1){
            System.out.println("given numbers are coprimes: ");
        }
        else{
            System.out.println("given numbers are not coprimes:");
        }
    }
}
```

Output

```
Enter the num 1 :
12
Enter the num2:
14
given numbers are not coprimes:
```

>>>>Check divisibility rules for numbers from 1 to 20.

Program

```
import java.util.Scanner;
class Main{
    public static void main(String[]args){
        Scanner sc =new Scanner(System.in);
        System.out.println("Enter the number 1");
        int num =sc.nextInt();
        for(int i =1;i<=20;i++){
            if(num%i==0)
            System.out.println("number is divisible by"+i);


            else{
               System.out.println("number is not divisible by"+i);
            }
            }


    }
}
```

Output

```
Enter the number 1
6
number is divisible by1
number is divisible by2
number is divisible by3
number is not divisible by4
number is not divisible by5
number is divisible by6
number is not divisible by7
number is not divisible by8
number is not divisible by9
number is not divisible by10
number is not divisible by11
number is not divisible by12
number is not divisible by13
number is not divisible by14
number is not divisible by15
number is not divisible by16
number is not divisible by17
number is not divisible by18
number is not divisible by19
```

number is not divisible by20

RECURSION:

>>>>> Some simple problems using recursion

 Programs :

```
class Main{
   public static int Sum(int n){
      if(n==0){
      return n;
      }
      return n +Sum(n-1);
   }
   public static void main(String[]args){
      int n = 10;
      int result = Sum(n);
      System.out.println(result);
   }
}
```

Output => 55

>>>print numbers from n to 1

```
class Main{
 public static void printreverseorder(int n){
      if(n==0){
      return ;
      }
      System.out.println(n +"");
      printreverseorder(n-1);
   }
   public static void main(String[]args){
      int n =6;
    printreverseorder(n);
    }
}
```

>>>>reverse of a number using recursion

```
class Main {
    public static String reverseString(String s) {
        if (s.isEmpty()) { // Base case
            return s;
        }
        return reverseString(s.substring(1)) + s.charAt(0); // Recursive case
    }

    public static void main(String[] args) {
        String str = "hello";
        System.out.println("Reversed string: " + reverseString(str)); // Output: olleh
    }
}
```

Output: olleh

>>>>palindrome using recursion

```
class Main{
    public static boolean  ispalindrome(String s,int start ,int end){
        if(start>=end){
        return true;
        }
        if(s.charAt(start)!=s.charAt(end)){
        return false;
        }
        return ispalindrome(s,start+1,end-1);

    }
    public static void main(String[]args){
        String s = "civic";
        boolean result = ispalindrome(s,0,s.length()-1);

        System.out.println("it is a palindrome : "+result);


}

}
```

Output -> it is a palindrome : true

>>>>> Count the digits of a given number using recursion..

Program,

```
class Main{
    public static int count(int num){
        if(num==0){
        return 0;
        }
        return 1+count(num/10);
    }
    public static void main(String[]args){
        int num = 3456;
        int result = count(num);
        System.out.println(result);
    }
}
```

Output :

```
4
```

Prime no.optimal approach

```
class Main {
    public static boolean isPrime(int n) {
        if (n <= 1) return false;
        for (int i = 2; i <= Math.sqrt(n); i++) {
            if (n % i == 0) return false;
        }
        return true;
    }


    public static void main(String[] args) {
        int num = 29;
        if (isPrime(num)) {
            System.out.println(num + " is a prime number.");
        } else {
            System.out.println(num + " is not a prime number.");
        }
    }
}    Output - 29 is a prime number.
```

>>>>find max element in array using recursion

```
class Main{
    public static int findmax(int[]arr,int n){
        if(n==1){
            return arr[0];
        }
        return Math.max(arr[n-1],findmax(arr,n-1));


    }
    public static void main(String[]args){
        int []arr ={1,2,3,5,4,6};
        int result = findmax(arr,arr.length);
        System.out.println(result);
    }
}
```

Output:6

>>>> program to add odd elements in the even indexes

Program

```
class Main{
    public static int sum(int[]arr){
        int sum =0;
        for(int i=0;i<arr.length;i++){
           if(i%2==0 && arr[i]%2!=0){
                  sum+=arr[i];
                }
        }
        return sum;
    }
    public static void main(String[]args){
        int []arr= {1,2,3,4,5,6,7,8};
        int res = sum(arr);
        System.out.println(res);
    }
}
```

Output :16

>>>>> gcd of a number using recursion

Program

```
class Main{
    public static int gcd(int a,int b){
        if(b==0){
        return a;
        }
        return gcd(b,a%b);
    }
    public static void main(String[]args){
        int a =43;
        int b =65;
        int res = gcd(a,b);
        System.out.println(res);
    }
}
```

Output : 1

□

>>>>>add all even numbers in descending order and odd numbers in ascending order and all should be placed in one array

Program

```
import java.util.Collections;
import java.util.ArrayList;
class Main{
    public static  void  solution(int[]arr){
        ArrayList<Integer>evenList = new ArrayList<>();
        ArrayList<Integer>oddList = new ArrayList<>();
        for(int num:arr){
            if(num%2==0){
              evenList.add(num);
            }
            else{
                oddList.add(num);
            }
        }
            Collections.sort(evenList,Collections.reverseOrder());
            Collections.sort(oddList);
            evenList.addAll(oddList);
              System.out.println(evenList);

        }
        public static void main(String[]args){
            int[]arr = { 1,3,2,4,5,6,7,8,9};
             solution(arr);
        }
    }
```

Output:

>>>> xor for n numbers

Bruteforce approach

```
class Main {
    public static void main(String[] args) {
        int n = 5;
        int result = 0;

        for (int i = 1; i <= n; i++) {
            result ^= i;
        }

        System.out.println("XOR of numbers from 1 to " + n + " is: " + result);
    }
}
```

Optimal approach :

```
class Main {
    public static void main(String[] args) {
        int n = 5;
        int result = xor(n);
        System.out.println("XOR of numbers from 1 to " + n + " is: " + result);
    }

    static int xor(int n) {
        if (n % 4 == 0) return n;
        if (n % 4 == 1) return 1;
        if (n % 4 == 2) return n + 1;
        return 0;
    }
}
```

Output : XOR of numbers from 1 to 5 is: 1

```
>>>sample input 1:abba
       Output : null
       Sample input 2:abbba
        Output : aba
```

Program

```
       import java.util.Stack;
class Main{
    public static String solution(String input){
        Stack<Character>s1 =new Stack<>();
        for(char ch:input.toCharArray()){
            if(!s1.isEmpty() && s1.peek() == ch){
                s1.pop();
            }
            else{
                s1.push(ch);
            }
        }
        StringBuilder result = new StringBuilder();
        for(char ch:s1){
            result.append(ch);
        }
      return  result.length() == 0?null:result.toString();
    }
    public static void main(String[]args){
        String input = "abba";
        String input2 ="abbba";
        System.out.println(solution(input));
        System.out.println(solution(input2));

    }
}
```

```
Ouput:null
            aba
```

Additional program :

```
import java.util.HashMap;

class Main {
    public static int solution(int[] arr) { // Pass the array as a parameter
        HashMap<Integer, Integer> map = new HashMap<>();

        for (int num : arr) {
            if (map.containsKey(num)) {
                map.remove(num); // Remove the number if it's already in the map
            } else {
                map.put(num, 1); // Add the number with a dummy value
            }
        }


        for (int key : map.keySet()) {
            return key; // Return the unique number
        }

        return -1;
    }

    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 2, 1};
        System.out.println(solution(arr));    }
}
// Output: 3
```

>>>>> unique elements should be printed

```
import java.util.HashMap;

class Main {
    public static int countUniqueElements(int[] nums) {
        HashMap<Integer, Integer> map = new HashMap<>();
        for (int num : nums) {
            map.put(num, map.getOrDefault(num, 0) + 1);
        }
        int uniqueCount = 0;
        for (int value : map.values()) {
            if (value == 1) {
                uniqueCount++;
            }
        }
        return uniqueCount;
    }

    public static void main(String[] args) {
        int[] nums = {1, 2, 3, 1, 4, 3, 4};
        int result = countUniqueElements(nums);
        System.out.println(result); // Output: 1 (Only "2" is unique)
    }
}
Output :1
```

Print even or odd numbers in a given range using recursion.

```
class Main{
    public static void count(int start,int end){
        if(start>end){
            return;
        }
        if(start%2==0){
            System.out.println(start + "even");
        }
        else if(start%2!=0){
            System.out.println(start + "odd");
        }
        count(start+1,end);
    }
    public static void main(String[]args){
        int start =1;
        int end = 5;
         count(start,end);
    }
}
```

```
o/p: 1odd
2even
3odd
4even
5odd

=== Code Execution Successful ===
```

```
public class Main {
    public static void main(String[] args) {
        printNumbers(1);
    }

    static void printNumbers(int n) {
        if (n > 5) return;
        System.out.println(n);
        printNumbers(n + 1);
    }
}
```

Output 1 2 3 4 5

Training sums continuation ……

Taget Output:5 4 3 2 1 0 1 2 3 4

```
public class Main {

    public static void main(String[] args) {

        printDescendingorder(5);

        printAscendingorder(1, 5);

    }
    public static void printDescendingorder(int n) {
        if (n < 0) {
            return;
        }
        System.out.println(n);
        printDescendingorder(n - 1);
    }
public static void printAscendingorder(int current, int max) {
        if (current == max) {
            return;
        }
        System.out.println(current);
        printAscendingorder(current + 1, max);
    }
}
5
4
3
2
1
0
1
2
3
4


=== Code Execution Successful ===
>> class Main{
    public static int factorial(int n){
        if(n==0){
            return 1 ;
        }
        return n*factorial(n-1);
    }
    public static void main(String[]args){
        int n =4;
        int res =factorial(n);
        System.out.println(res);
    }
}
```

24

```
class Main {
    static int fibonacci(int n) {
        if (n <= 1) {
            return n;
        }
        return fibonacci(n - 1) + fibonacci(n - 2);
    }

    public static void main(String[] args) {
        System.out.println(fibonacci(9));
    }
}
```

Output: 34

Program to find Armstrong number using recursion

```
class Main {
    public static int countDigits(int num) {
        if (num == 0) {
            return 0;
        }
        return 1 + countDigits(num / 10);
    }
    public static boolean isArmstrong(int num) {
        int originalNum = num;
        int numOfDigits = countDigits(num);
        int sum = 0;

        while (num != 0) {
            int digit = num % 10;
            sum += Math.pow(digit, numOfDigits);
            num /= 10;
        }

        return sum == originalNum;
    }
    public static void main(String[] args) {
        int num = 153;
        if (isArmstrong(num)) {
            System.out.println(num + " is an Armstrong number.");
        } else {
            System.out.println(num + " is not an Armstrong number.");
        }
    }
}
```

Output :

>>>> sum of digits of a number using recursion

```
class Main {
    public static int sumDigits(int num) {
        if (num == 0) {
            return 0;
        }
        return num % 10 + sumDigits(num / 10);
    }

    public static void main(String[] args) {
        int num = 23456;
        System.out.println(sumDigits(num));
    }
}
class Main {
    public static void printreverseorder(int n) {
        if (n == 0) {
            return;
        }
        printreverseorder(n - 1);
        System.out.println(n);
        System.out.println("200");
    }

    public static void main(String[] args) {
        int n = 5;
        printreverseorder(n);
    }
}
```

Output

```
1
200
2
```

```
200

3

200

4

200

5

200
```

Note ; the actual intention of the above code is to print numbers in reverse order and include 200 in between every number but due to misplacement of printreverseorder(n-1) it prints numbers in ascending order instead of descending

Now have a look on the next code where printreverseorder(n-1)is placed perfectly

```
class Main {
    public static void printreverseorder(int n) {
        if (n == 0) {
            return;
        }
        System.out.println(n);
        System.out.println("200");
       printreverseorder(n - 1);


    }


    public static void main(String[] args) {
        int n = 5;
        printreverseorder(n);
    }
}
```

Output

```
5

200

4

200
```

```
3

200

2

200

1

200
```

>>> odd numbers upto 100 using recursion

```
public class Main {
    public static void printOddNumbers(int n) {
        if (n >= 100) {
            return;
        }
        if (n % 2 != 0) {
            System.out.println(n);
        }
        printOddNumbers(n + 1);
    }

    public static void main(String[] args) {
        int start = 1;
        printOddNumbers(start);
    }
}
```

Output

Odd numbers upto 100 will be displayed

>>>   Same case for even numbers:

```
public class Main {
    public static void printOddNumbers(int n) {
        if (n >= 100) {
            return;
        }
        if (n % 2 == 0) {
            System.out.println(n);
        }
        printOddNumbers(n + 1);
    }

    public static void main(String[] args) {
        int start = 1; // Start printing odd numbers from 1
        printOddNumbers(start);
    }
}
```

>>>>decimal to binary conversion

```
public class Main {
    public static void main(String[] args) {
        int decimalNumber = 19;
        StringBuilder binary = new StringBuilder();

        while (decimalNumber > 0) {
            binary.append(decimalNumber % 2); // Append remainder (binary digit)
            decimalNumber /= 2; // Divide the number by 2
        }

        // Reverse the string to get the correct binary representation
        System.out.println(binary.reverse().toString());
    }
}
```

```
10011
```

>>>>print elements in array using recursion

```
class Main {
    public static int solution(int[] arr) {
        if (arr.length == 0) {
            return 0; // If the array is empty, return 0
        }
        return arr.length;
    }

    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        System.out.println(solution(arr));
    }
}
```

Op-5

>>>>CODEFORCES – 1560/A

memory limit per test256 megabytes

Polycarp doesn't like integers that are divisible by 3

or end with the digit 3

in their decimal representation. Integers that meet both conditions are disliked by Polycarp, too.

Polycarp starts to write out the positive (greater than 0

) integers which he likes: 1,2,4,5,7,8,10,11,14,16,…

. Output the k

-th element of this sequence (the elements are numbered from 1

).

Input

The first line contains one integer t

(1≤t≤100

) — the number of test cases. Then t

test cases follow.

Each test case consists of one line containing one integer k

(1≤k≤1000

).

Output

For each test case, output in a separate line one integer x

— the k

-th element of the sequence that was written out by Polycarp.

Example

InputCopy

10

1

2

3

4

5

6

7

8

9

1000

OutputCopy

1

2

4

5

7

8

```
10
11
14
1666
```

```
class Main {
    public static int solution(int k) {
        int count = 0;
        int current = 0;

        while (true) {
            current++;
            if (current % 3 != 0 && current % 10 != 3) {
                count++;

            if (count == k) {
                return current;
            }
        }
    }
}

    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5, 6, 7, 8, 9, 1000};
        for (int k : arr) {
            System.out.println(solution(k));
        }
    }
}
```

Output :

```
1
2
```

```
4
5
7
8
10
11
14
1666
```

>>>> print elements of the array using recursion

```
public class Main {
   public static void printArray(int[] arr, int index) {
       if (arr == null || index >= arr.length) {
           return;
       }
       System.out.println(arr[index]);
       printArray(arr, index + 1);
   }
   public static void main(String[] args) {
       int[] arr1 = {1, 2, 3, 4, 5};
       int[]arr2 ={};
       int[] arr3 = {42};

       System.out.println("Array 1:");
       printArray(arr1, 0);

       System.out.println("Array 2:");
       printArray(arr2, 0);

       System.out.println("Array 3:");
       printArray(arr3, 0);
   }
}
```

```
Output Array 1:
1
2
3
4
5
Array 2:
Array 3:
42
```