1. Kadane Algo for Max sum of Subarray in a Array.

```
class Kadane
{
	static int maxSubarray(int nums[])
	{
		int max = nums[0];
		int sum = 0;
		for(int i=0;i<nums.length;i++){
			sum+=nums[i];

			if(sum > max)max = sum;

			if(sum <0) sum =0;
		}
		return max;
	}
	public static void main(String[] args) {
		int nums[]={-2,-3,4,-1,-2,1,5,-3};
		System.out.print("The  maxSubarray sum is : " +maxSubarray(nums));
	}
}
```

```
// Day 1.5 Merge The Intervals

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

```
// Optimized Method By using stack
        public static void merge_interval(Interval arr[]){
    if(arr.length <= 0)
        return;
    Stack<Interval> stk =new Stack<>();
    Arrays.sort(arr,new Comparator<Interval>(){
        public int compare(Interval i1, Interval i2){
                return i1.start - i2.start;
        }
    });
    stk.push(arr[0]);
    for (int i=1;i<arr.length ;i++ ) {

        Interval top = stk.peek();
        if( top.end <arr[i].start){
                stk.push(arr[i]);
        }
        else if(top.end<arr[i].end){
                top.end= arr[i].end;
                stk.pop();
                stk.push(top);
        }
    }
    System.out.println("The Merged Intervals are  :" );
    while(!stk.isEmpty())
    {
        Interval t = stk.pop();
        System.out.print("[" +t.start+","+ t.end + "]");
```

```
    }
        }
    public static void main(String[] args) {
      Interval arr[]=new Interval[4];
      arr[0]=new Interval(1,3);
      arr[1]=new Interval(2,6);
      arr[2]=new Interval(15,19);
      arr[3]=new Interval(20,22);
      merge_interval(arr);
    }
}
class Interval{
        int start,end;
        Interval(int start,int end)
        {
                this.start=start;
                this.end=end;
        }
}


// Insertion sort having the TIME COMPLEXITY O(N*N)
import java.util.*;
class insertion {

        static void print_sortList(int a[]){
                for (int i=0;i<a.length ;i++ ) {
                        System.out.print(a[i]+" ");
                }
```

```
	}
	static void insertion_sort(int a[]){
		for(int i=1 ;i<a.length;i++){

			int key = a[i];
			int j = i-1;
			while(j > -1 && a[j] > key){
				a[j+1]=a[j];
				j--;
			}
			a[j+1] = key;
			System.out.print("Iteration"+i+"= ");
			print_sortList(a);
			System.out.println();
		}
		 print_sortList(a);

	}
		public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		int n = sc.nextInt();
		int a[]=new int[n];
		for (int i=0;i<n ;i++ ) {

			a[i]=sc.nextInt();
		}
		insertion_sort(a);
	}
}
```

```
// Day 1.6 Find Duplicate Number for i+1 size
import java.util.*;

class duplicateno
{

  // Solve By using Hashmap  O(n) and  space :O(n)
        static int duplicate(int arr[])
        {

                Map<Integer,Boolean> mp =new HashMap<>();

                for(Integer i : arr){

                        if(mp.get(i) == null){
                                mp.put(i,true);
                        }
                        else{
                                return i;
                        }
                }
                return 0;

        }

        // solve by using tortoize linked list tortoise method

        static int duplicate1(int arr[])
```

```
	{

		int fast=0;
		int slow=0;
		do{

		 slow=arr[slow];
		 fast=arr[arr[fast]];
		}
		while(slow!=fast);



		 fast = 0;
		 while(slow!=fast)
		 {
			slow=arr[slow];
			fast=arr[fast];
		 }
		 return slow;

	}
	public static void main(String[] args) {
		int arr[] ={1,2,3,4,5,6,3};
		// System.out.println("The duplicate No is :" + duplicate(arr));
		System.out.println("The duplicate No is :" + duplicate1(arr));
	}
}
```

```
// Day 1.4
 import java.util.*;



 class MergeTwoSortedArray{


  static int nextGap(int gap){
          if(gap<= 1){
                  return 0;
          }
          else{
           return (gap/2) + (gap%2);
          }
  }


  static void MergeTwoSortedArray_m(int arr1[],int arr2[],int n,int m){



    int i,j, gap=n+m;
    for(gap=nextGap(gap) ; gap > 0 ;gap=nextGap(gap)){
      // For First Array
      for(i=0;i+gap<n;i++){
          if(arr1[i]> arr1[i+gap]){
                  int temp=arr1[i];
                  arr1[i]=arr1[i+gap];
                  arr1[i+gap]=temp;
          }
      }
```

```
        // for Both Arrays

        for(j =gap > n ? gap - n:0; i<n && j<m ; i++,j++){
            if(arr1[i]> arr2[j]){
                        int temp=arr1[i];
                        arr1[i]=arr2[j];
                        arr2[j]=temp;
            }
        }
        // for Second Array

         if (j<m){
            for (j=0;j+gap<m ;j++ ) {
                        if(arr2[j]>arr2[j+gap]){
                        int temp=arr2[j];
                        arr2[j]=arr2[j+gap];
                        arr2[j+gap]=temp;
                        }
            }
         }


      }

    }

// Another  Method i.e Insertion Method for Merging Bt I requires O(n*m)
   static void merge(int arr1[],int arr2[],int n ,int m)
   {
```

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

        if(arr1[i] > arr2[0])
        {
                int temp = arr1[i];
                arr1[i]= arr2[0];
                arr2[0]=temp;
        }
        Arrays.sort(arr2);
 }

 }


public static void main(String[] args){
 int arr1[]={1,3,5,6,7};
 int arr2[]={2,4,8};
 int n=arr1.length;
 int m=arr2.length;
 //MergeTwoSortedArray_m(arr1,arr2,n,m);

 merge(arr1,arr2,n,m);

 System.out.println("The First Array :");
 for ( int i=0;i<n ;i++ ) {
        System.out.print(arr1[i] + " ");
 }
 System.out.println("\nThe Second Array :");
 for ( int i=0;i<m ;i++ ) {
```

```
                System.out.print(arr2[i] + " ");
            }
        }
}


// Day1.2 Find the Missing and Repeating Number
import java.util.*;

 class MissingandRepeting{
        // Using HashMap O(n)+O(n)
        static void find_m_r(int nums[])
        {

                int n=nums.length;
                Map<Integer,Boolean> numberMap = new HashMap<>();
                for(Integer i: nums){
                        if(numberMap.get(i)==null)
                        {
                                numberMap.put(i,true);
                        }
                        else
                        {
                                System.out.println("Repeating :" + i);
                        }
                }
                for(int i=0;i<n;i++)
                {
                        if(numberMap.get(i)==null)
```

```
                        System.out.println("Missing :" + i);
                }

        }

        // Optimized and Faster Method Using XOR operator
static int x,y;
        static void find_m_r1(int arr[])
        {
                int n=arr.length;
                int xor1=arr[0];
                int set_bit_no;
                x=0;
                y=0;
                int i;

                for( i=1;i<n;i++)
                {
                        xor1=xor1^ arr[i];
                }
                System.out.println(xor1);

                for( i=1;i<=n;i++)
                {
                        xor1=xor1^ i ;
                }
                System.out.println(xor1);

                set_bit_no = xor1 & ~(xor1 -1 );
```

```
        System.out.println(set_bit_no);


        for( i=0;i<n;i++)
        {
                if((arr[i] & set_bit_no )!=0){
                        x= x^ arr[i];
                }
                else{
                        y=y^ arr[i];
                }
        }
        for(i=1;i<=n;i++)
        {
                if((i & set_bit_no )!=0){
                        x= x^ i;
                }
                else{
                        y=y^ i;
                }
        }




        System.out.println("The missing element is"+x+"and the "+"repeating number is "+y);

}

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

```
                //find_m_r(nums);
                find_m_r1(nums);
        }
}

import java.util.*;

class pattern1{
        public static void main(String[] args) {
                Scanner sc= new Scanner(System.in);
                int n = sc.nextInt();
                int k,j;
                for (int i=1; i<= n ;i++ ) {
                        for (j=1; j<=n-i;j++ ) {
                                System.out.print(" ");
                        }
                        k=i;
                        for (;j<=n ;j++ ) {

                                System.out.print(k--);
                        }
                        k=2;
                        for (;j<= n+i-1 ;j++) {
                           System.out.print(k++);

                        }
                System.out.println();
                }
        }}
```

```
// selection sort  having the TIME COMPLEXITY O(N*N)
import java.util.*;

class selection{

        static void print_sortList(int a[]){
                for (int i=0;i<a.length ;i++ ) {
                        System.out.print(a[i]+" ");
                }
        }
  static void swap(int a[],int i,int j){
   int tmp = a[i];
   a[i]=a[j];
   a[j]=tmp;
  }
        static void selection_sort(int a[],int l){
                for(int i=0;i<l;i++){
                        int min = i;
                        for (int j = i+1; j< l ;j++ ){
                                if(a[j]<a[min]){
                                        min = j;


                                }


                        }
                        swap(a,i,min);
                        System.out.print("Iteration"+i+"= ");
                         print_sortList(a);
                         System.out.println();
```

```
                }
                print_sortList(a);
        }
        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                int n = sc.nextInt();
                int a[]=new int[n];
                int len = a.length;
                for (int i=0;i<n ;i++ ) {

                        a[i]=sc.nextInt();
                }
                selection_sort(a,len);
        }
}




// dutch  national flag algo..

// Day1.1 Problem of sort an array of 0,1 and 2's
// Time complexity = O(N)
// Space complexity = O(1)


class sort012{

   static void sort_m(int nums[])
```

```
{
        int lo=nums[0];
        int mid=nums[0];
        int high=nums.length-1;
        int temp;

        while(mid<=high)
        {
                switch(nums[mid])
                {
                        case 0:{
                                temp=nums[lo];
                                nums[lo]=nums[mid];
                                nums[mid]=temp;

                                mid++;
                                lo++;
                                break;
                            }
                        case 1:
                              mid++;
                                        break;
                        case 2: temp=nums[high];
                              nums[high]=nums[mid];
                              nums[mid]=temp;
                              high--;

                              break;
                }
```

```
        }

        System.out.println("Sorted Array is:");
        for(int n=0;n<nums.length;n++){
                System.out.print(nums[n]);
        }

    }
        public static void main(String[] args) {
                int nums[]={0,0,2,1,2,0,2,1,1,0,1};
                sort_m(nums); import java.util.*;
// this is program in which we swap the numbers without using 3rd varible

class swaps{

static void swap(int a,int b){
        a=a^b;
        //System.out.println(a);
        b=b^a;
        a=a^b;
        System.out.println("Numbers  After Swaps : a=" + a + " b="+ b);
}
static void swap1(int a ,int b,int arr[]){
        a= a+b;
        b=a-b;
        a=a-b;
        System.out.println("Numbers  After Swaps : a=" + a + " b="+ b);
```

```
        int xor1=arr[0];
        for (int i=1;i<arr.length;i++ ) {
                System.out.print(xor1 + "^" +arr[i]+"=");
                xor1^=arr[i];
                System.out.print(xor1);
                System.out.println();
        }
        System.out.println(xor1);



}
        public static void main(String[] args) {

                int a,b;
                Scanner sc = new Scanner(System.in);
                System.out.println("ENter the numbers");
                a=sc.nextInt();
                b=sc.nextInt();
                System.out.println("Numbers  Before Swaps : a=" + a + " b="+ b);
                int a1[]={3,4,5,2,1,1};
                swap1(a,b,a1);


        }
}


// Permutation of a Number
```

```
import java.util.*;
class permutation{
   static List<List<Integer>> permute(int a[]){
                List<List<Integer>> res =  new ArrayList<>();
                boolean visited[] = new boolean[a.length];
                List<Integer> curr = new ArrayList<>();
                backtrack(res,a,curr,visited);
                return res;
   }

   static void backtrack(List<List<Integer>> res, int a[],List<Integer> curr, boolean visited[]){
          if(curr.size() == a.length){
                    res.add(new ArrayList(curr));
                    return;
          }

          for (int i=0;i<a.length;i++ ) {
                    if(visited[i]==true) continue;
                    curr.add(a[i]);
                    visited[i]=true;
                    backtrack(res,a,curr,visited);
                    curr.remove(curr.size()-1);
                    visited[i]=false;

          }

   }
```

```
   static List<List<Character>> permute_str(String str){
        List<List<Character>> res1 = new ArrayList<>();
      boolean[] visited = new boolean[str.length()];
      List<Character> curr_s = new ArrayList<>();
      backtrack_str(res1,str,curr_s,visited);
      return res1;
   }

    static void backtrack_str(List<List<Character>> res1, String str, List<Character> curr_s,boolean[]
visited){

        if(curr_s.size() == str.length()){
                res1.add(new ArrayList(curr_s));
                return;
        }
        for (int i=0;i<str.length() ;i++ ) {
                if(visited[i]==true)continue;
                curr_s.add(str.charAt(i));
                visited[i]=true;
                backtrack_str(res1,str,curr_s,visited);
                curr_s.remove(curr_s.size()-1);
                visited[i]=false;



        }


    }

        public static void main(String[] args) {
```

```
                // Print All permutation of Number
                /*int arr[] ={1,2,3};
                System.out.println(permute(arr));          */
    // Print permuation of ALL String

                String str ="abc";
                System.out.println(permute_str(str));



        }

}

// Rainwater Trapped Problem
// Optimized Approach Time Complexity : O(n) && Space Complexity : O(1)
import java.util.*;
class rainwater{
        static int rainwater(int a[]){
                int n = a.length;
                int max=0;
                int leftmax=0;
                int water=0;
                int maxindex=0;
                for (int i=0; i<n;i++ ) {
                        if(max < a[i])
                        {
                         max = a[i];
                         maxindex=i;
                        }
```

```
                }
                for (int i=0;i<maxindex ;i++ ) {
                        leftmax=Math.max(leftmax,a[i]);
                        int w=Math.min(leftmax,max)-a[i];
                        water+=w;
                }
                leftmax=0;
                for(int i=n-1;i>=maxindex;i--){
                        leftmax = Math.max(leftmax,a[i]);
                        int w= Math.min(leftmax,max)-a[i];
                        water+=w;
                }
                return water;
        }
                public static void main(String[] args) {
                int arr[] = {0,1,2,1,0,3,1,2,1};
                int total_water =rainwater(arr);
                System.out.println("The total_water Gained :" + total_water);
        }
}

// Rotation in circular array
// time complexity is O(logn)

class rotation_circularArray{

        static int rotation_carray(int a[],int l, int h){

                if (l>h) return 0;
```

```
		if(l==h) return l;

		int mid = l + (h-l) /2 ;
		//check if mid is greater
		if (mid < h  && a[mid+1] < a[mid] ) {
			return mid+1;
		}
		// check for smaller mid

		if (mid > l && a[mid] < a[mid-1])  {
			return mid;
		}

		if (a[h] > a[mid]) {

			return rotation_carray(a,l,mid-1);

		}
		return rotation_carray(a, mid+1, h);
	}

   public static void main(String[] args) {
	int arr[] ={ 12,13, 1,2,3,15};

	System.out.println("The total Rotation is  :" + rotation_carray(arr,0,arr.length-1));

   }
}
```