Name: Devesh

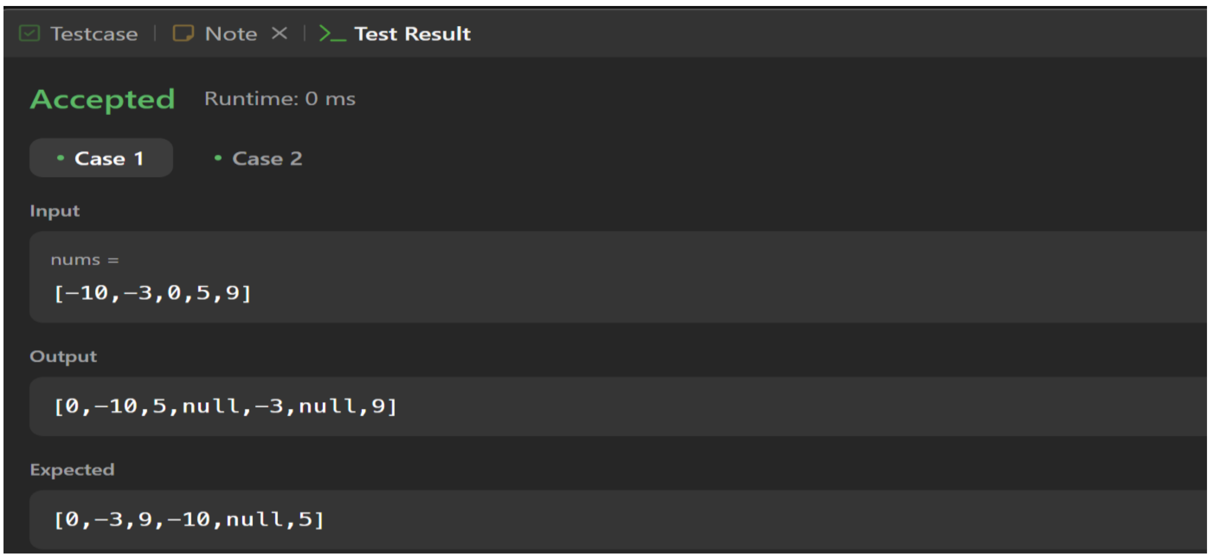
UID: 22BCS16690

Class: 605 - B

Q 1 Convert Sorted Array to Binary Search Tree

```
class Solution {
    public TreeNode createBST(int nums[] , int x , int y){
        if(x>y){
            return null;
        }
        int mid = (x+y)/2;
        TreeNode root = new TreeNode(nums[mid]);
        root.left = createBST(nums,x,mid-1);
        root.right = createBST(nums,mid+1,y);
        return root;
    }
    public TreeNode sortedArrayToBST(int[] nums) {
        return createBST(nums , 0 , nums.length-1);
    }
}
```

OUTPUT:



Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

```
nums =
[-10,-3,0,5,9]
```

Output

```
[0,-10,5,null,-3,null,9]
```

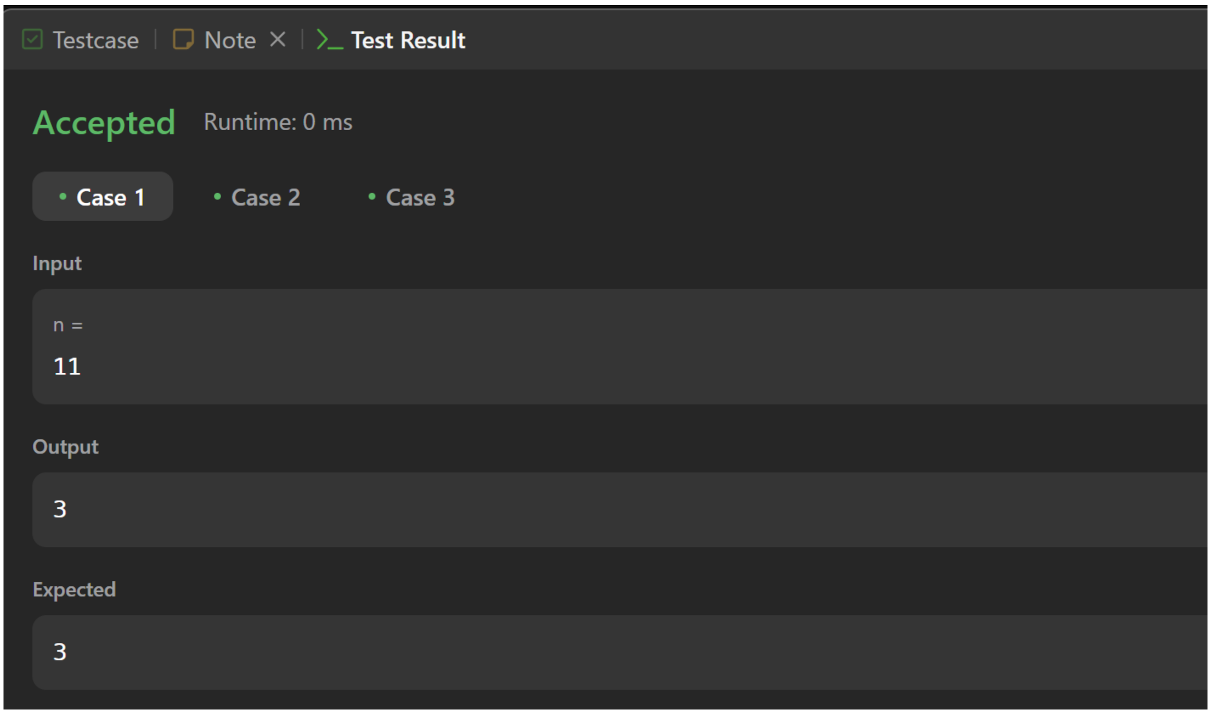
Expected

```
[0,-3,9,-10,null,5]
```

Q 2 Number of 1 Bits

```
class Solution {
    public int hammingWeight(int n) {
        int count = 0;
        while(n>0){
            if((n & 1) != 0){
                count++;
            }
            n = n>>1;
        }
        return count;
    }
}
```

OUTPUT:



```
Accepted   Runtime: 0 ms

Case 1   Case 2   Case 3

Input
n =
11

Output
3

Expected
3
```

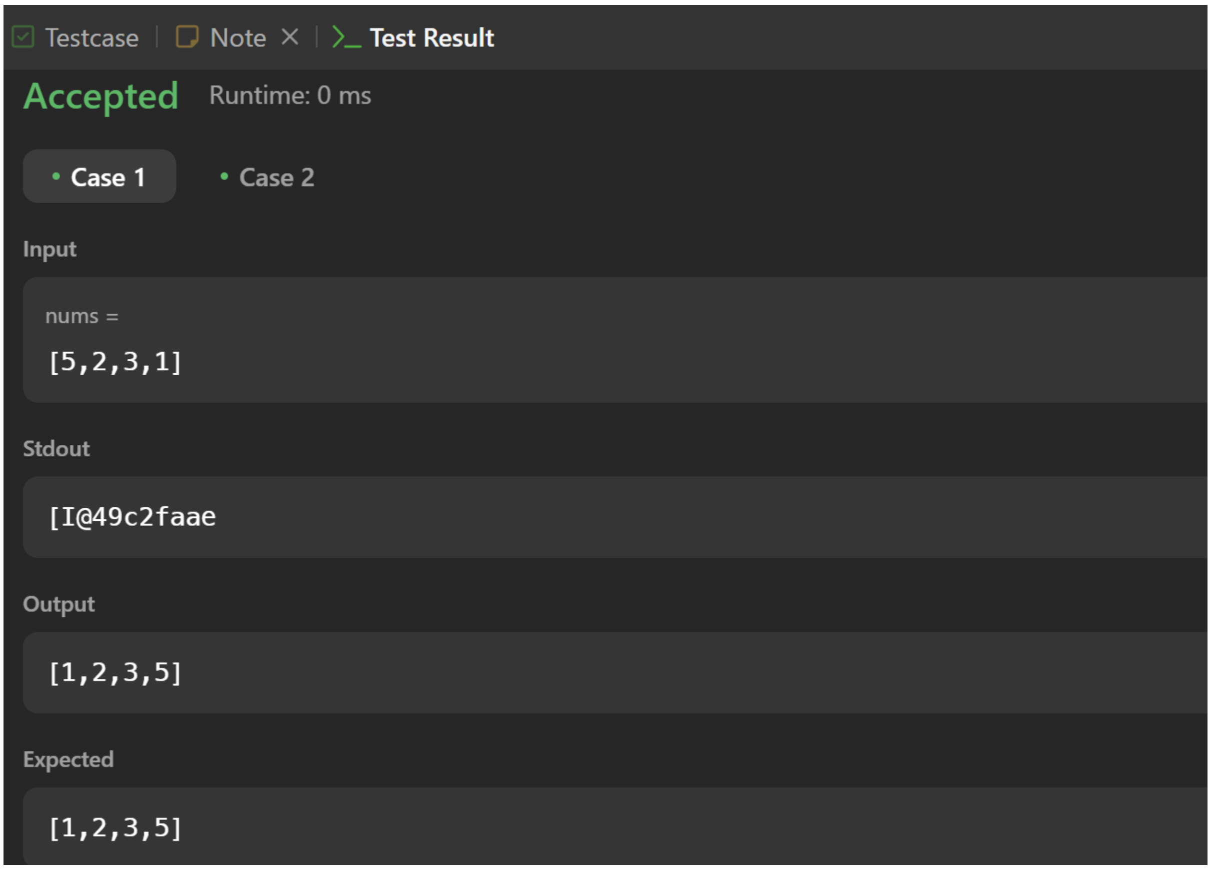
Q3 Sort an Array

```
class Solution {
    public int[] sortArray(int[] nums) {
        nums = mergeSort(nums);
        System.out.print(nums);
        return nums;
    }
    private int[] mergeSort(int[] nums){
        if(nums.length == 1){
            return nums;
        }
        int mid  = nums.length/2;
        int[] left = mergeSort(Arrays.copyOfRange(nums,0,mid));
        int[] right = mergeSort(Arrays.copyOfRange(nums,mid,nums.length));
        return merge(left,right);
    }
    private int[] merge(int[] first ,int[] second){
        int[] mix = new int[first.length + second.length];
        int  i = 0;
        int j = 0;
        int k = 0;
        while(i<first.length && j<second.length){
            if(first[i]<second[j]){
                mix[k] = first[i];
                i++;
            }
            else{
                mix[k] = second[j];
                j++;
            }
            k++;
        }
        while(i<first.length ){

                mix[k] = first[i];
                i++;
            k++;
          }
          while( j<second.length){

            mix[k] = second[j];
            j++;
            k++;
          }
          return mix;
    }
}
```

OUTPUT:



```
Accepted    Runtime: 0 ms

Case 1    Case 2

Input
nums =
[5,2,3,1]

Stdout
[I@49c2faae

Output
[1,2,3,5]

Expected
[1,2,3,5]
```

Q 4 Maximum Subarray

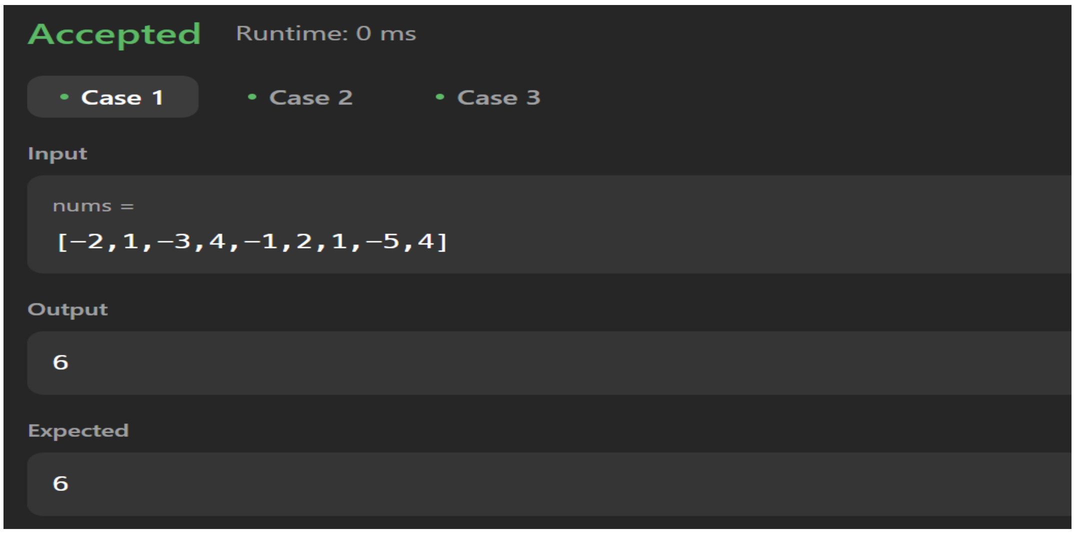
```
class Solution {
    public int maxSubArray(int[] nums) {
        int maxSum = Integer.MIN_VALUE;
        int currentSum = 0;

        for (int i = 0; i < nums.length; i++) {
            currentSum += nums[i];

            if (currentSum > maxSum) {
                maxSum = currentSum;
            }

            if (currentSum < 0) {
                currentSum = 0;
            }
        }
        return maxSum;
    }
}
```

OUTPUT:

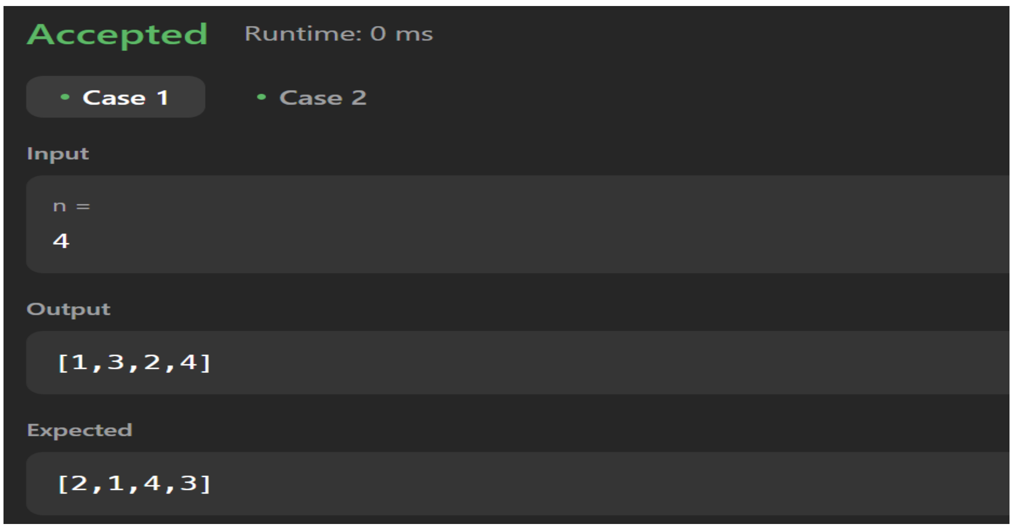


```
Accepted   Runtime: 0 ms
• Case 1   • Case 2   • Case 3
Input
nums =
[-2,1,-3,4,-1,2,1,-5,4]
Output
6
Expected
6
```

Q 5 Beautiful Array

```
class Solution {
    public int[] beautifulArray(int N) {
        int[] res = new int[N];
        if (N == 1)
        {
           return new int[] {1};
        }
        else if (N == 2)
        {
           return new int[] {1, 2};
        }
        else
        {
           int[] odds = beautifulArray((N + 1) / 2);
           int[] even = beautifulArray(N / 2);
           for (int i = 0; i < odds.length; i ++)
           {
              res[i] = odds[i] * 2 - 1;
           }
           for (int j = 0; j < even.length; j ++)
           {
              res[odds.length + j] = even[j] * 2;
           }
        }
        return res;
     } }
```

OUTPUT:

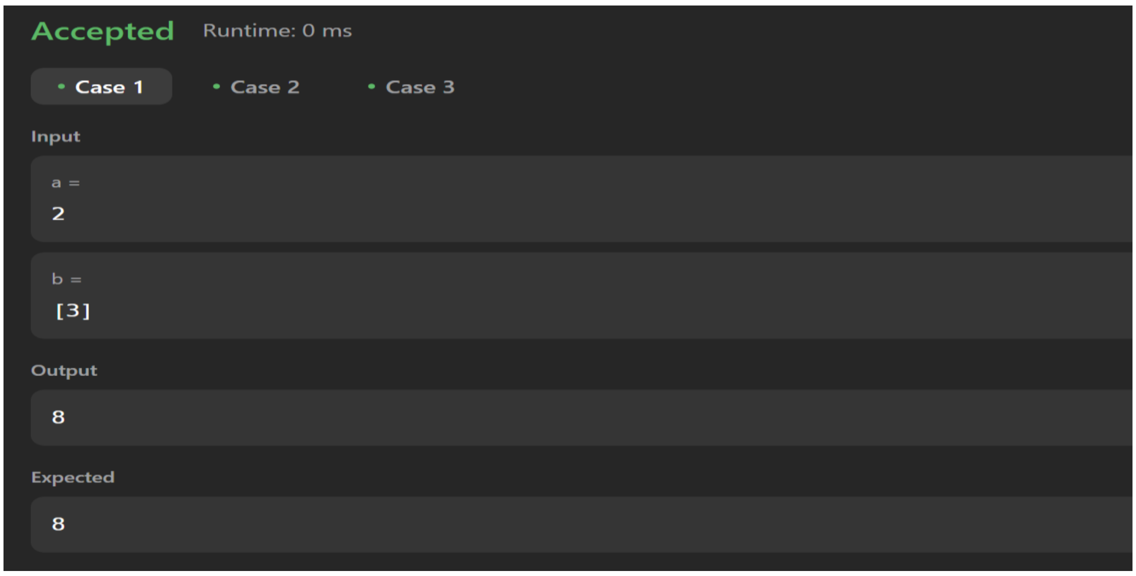


```
Accepted    Runtime: 0 ms
• Case 1    • Case 2
Input
n =
4
Output
[1,3,2,4]
Expected
[2,1,4,3]
```

Q 6 Super Pow

```
class Solution {
   public int superPow(int a, int[] b) {
      int num=0;
      for(int i:b){
         num=(num*10+i)%1140;
      }
      return binexpo(a,num,1337);
   }
   public int binexpo(int a, int b, int m){
      a%=m;
      int res=1;
      while(b>0){
         if((b&1)==1)
            res=(res*a)%m;
         a=(a*a)%m;
         b>>=1;
      }
      return res;
   }
}
```

OUTPUT:



```
Accepted    Runtime: 0 ms

• Case 1    • Case 2    • Case 3

Input
a =
2
b =
[3]
Output
8
Expected
8
```

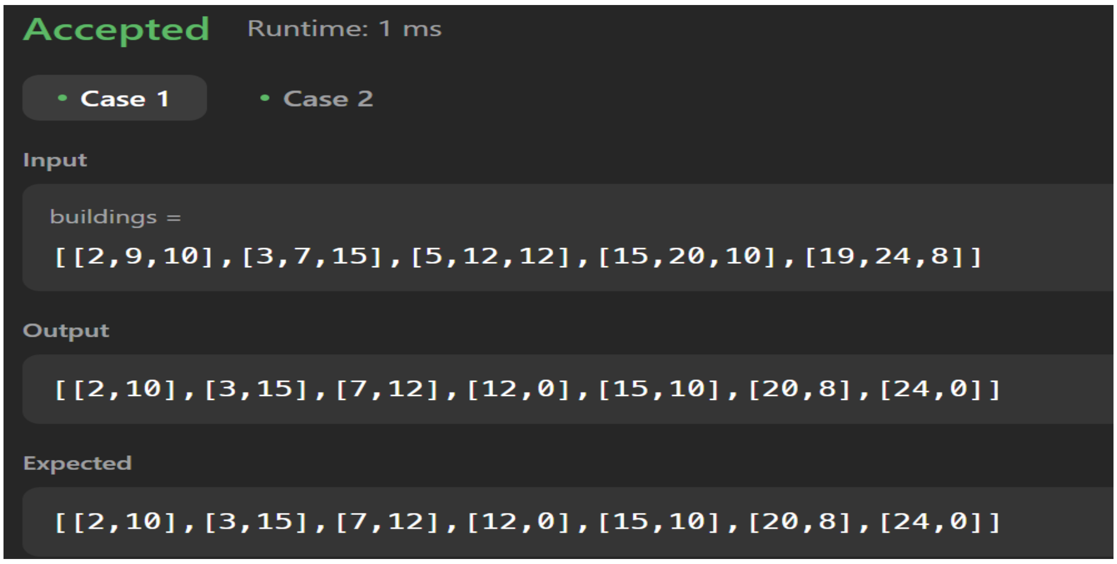
Q 7 The Skyline Problem

```
class Solution {
public List<List<Integer>> getSkyline(int[][] buildings) {
   List<List<Integer>> list = new ArrayList<>();


   List<int[]> lines = new ArrayList<>();
   for (int[] building: buildings) {
      lines.add(new int[] {building[0], building[2]});
      lines.add(new int[] {building[1], -building[2]});
   }
   Collections.sort(lines, (a, b)->a[0]==b[0]?b[1]-a[1]:a[0]-b[0]);
   TreeMap<Integer, Integer> map = new TreeMap<>();
   map.put(0, 1);
   int prev=0;
   for (int[] line: lines) {
      if (line[1]>0) {
         map.put(line[1], map.getOrDefault(line[1], 0)+1);
      } else {
         int f = map.get(-line[1]);
         if (f==1) map.remove(-line[1]);
         else map.put(-line[1], f-1);
      }
      int curr = map.lastKey();
      if (curr!=prev) {
         list.add(Arrays.asList(line[0], curr));
         prev=curr;
      }
   }
   return list;
```

```
    }
}
```

OUTPUT:



Accepted Runtime: 1 ms

Input

```
buildings =
[[2,9,10],[3,7,15],[5,12,12],[15,20,10],[19,24,8]]
```

Output

```
[[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]
```

Expected

```
[[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]
```