# Assignment 6

## Student Name: Anuj Kumar UID:22BCS11581

## Branch: BE-CSE (General) Section/Group: FL\_IOT-602-A

## Semester:6th Date of Performance: 18-03-25

## Subject Name: Advanced Programming Lab-2 Subject Code: 22CSP-351

# Aim: [108. Convert Sorted Array to Binary Search Tree](https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree/)

# Implementation/ Code:

# class Solution {

# public TreeNode sortedArrayToBST(int[] nums) {

# return constructBST(nums, 0, nums.length - 1);

# }

# private TreeNode constructBST(int[] nums, int left, int right) {

# if (left > right) {

# return null;

# }

# 

# int mid = left + (right - left) / 2;

# TreeNode root = new TreeNode(nums[mid]);

# root.left = constructBST(nums, left, mid - 1);

# root.right = constructBST(nums, mid + 1, right);

# 

# return root;

# }

# }

# Output:

# 

# Aim: [191. Number of 1 Bits](https://leetcode.com/problems/number-of-1-bits/)

# Implementation/ Code:

# class Solution {

# public int hammingWeight(int n) {

# int count = 0;

# while (n>0) {

# n &= (n - 1);

# count++;

# }

# return count;

# }

# }

# Output:

# Aim: [912. Sort an Array](https://leetcode.com/problems/sort-an-array/)

# Implementation/ Code:

# class Solution {

# public int[] sortArray(int[] nums) {

# PriorityQueue<Integer> pq = new PriorityQueue<>();

# for(int i : nums){

# pq.add(i);

# }

# int i =0;

# while(!pq.isEmpty()){

# nums[i++] = pq.poll();

# }

# return nums;

# }

# }

# Output:

# 

# Aim: [53. Maximum Subarray](https://leetcode.com/problems/maximum-subarray/)

# Implementation/ Code:

# class Solution {

# public int maxSubArray(int[] nums) {

# int maxSum = nums[0], currentSum = nums[0];

# for (int i = 1; i < nums.length; i++) {

# currentSum = Math.max(nums[i], currentSum + nums[i]);

# maxSum = Math.max(maxSum, currentSum);

# }

# return maxSum;

# }

# }

# Output:

# Aim:[932. Beautiful Array](https://leetcode.com/problems/beautiful-array/)

# Implementation/ Code:

# class Solution {

# public int[] beautifulArray(int n) {

# int[] answer = new int[n];

# if(n == 1) {

# answer[0] = 1;

# return answer;

# }

# int[] right =beautifulArray(n/2);

# int[] left = beautifulArray((n+1)/2);

# 

# for(int i=left.length; i<n; i++) { //This loop adds all even elements at end

# answer[i] = right[i-left.length] \* 2;

# }

# for(int i=0; i<left.length; i++) { //This loop adds all odd elements at start

# answer[i] = left[i] \* 2 - 1;

# }

# return answer;

# }

# }

# Output:

# 

# Aim: [372. Super Pow](https://leetcode.com/problems/super-pow/)

# Implementation/ Code:

# class Solution {

# private final int MOD = 1337;

# public int superPow(int a, int[] b) {

# return helper(a, b);

# }

# private int helper(int a, int[] b) {

# int res = 1;

# for (int i = b.length-1; i >= 0; i--) {

# res = (int) (res \* pow(a, b[i])) % MOD;

# a = (int) pow(a, 10);

# }

# return res;

# }

# private int pow(int a, int b) {

# int res = 1;

# a %= MOD;

# for (int i = 0; i < b; i++) {

# res = (res \* a) % MOD;

# }

# return res;

# }

# }

# Output:

# Aim: [218. The Skyline Problem](https://leetcode.com/problems/the-skyline-problem/)

# Implementation/ Code:

# public List<int[]> getSkyline(int[][] buildings) {

# List<int[]> result = new ArrayList<>();

# List<int[]> height = new ArrayList<>();

# for(int[] b:buildings) {

# height.add(new int[]{b[0], -b[2]});

# height.add(new int[]{b[1], b[2]});

# }

# Collections.sort(height, (a, b) -> {

# if(a[0] != b[0])

# return a[0] - b[0];

# return a[1] - b[1];

# });

# Queue<Integer> pq = new PriorityQueue<>((a, b) -> (b - a));

# pq.offer(0);

# int prev = 0;

# for(int[] h:height) {

# if(h[1] < 0) {

# pq.offer(-h[1]);

# } else {

# pq.remove(h[1]);

# }

# int cur = pq.peek();

# if(prev != cur) {

# result.add(new int[]{h[0], cur});

# prev = cur;

# }

# }

# return result;

# }

# Output:

# 