**DSA TRAINING PRACTICE SET Sharavana kumar S(22AD122) AIDS**

1. 1. Maximum Subarray Sum – Kadane‟s Algorithm:

Given an array arr[], the task is to find the subarray that has the maximum sum and return its

sum.

Input: arr[] = {2, 3, -8, 7, -1, 2, 3}

Output: 11

Explanation: The subarray {7, -1, 2, 3} has the largest sum 11.

Input: arr[] = {-2, -4}

Output: –2

Explanation: The subarray {-2} has the largest sum -2.

Code:

import java.util.\*;

public class One {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int n = scanner.nextInt();

ArrayList<Integer> nums = new ArrayList<>();

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

nums.add(scanner.nextInt());

}

int maxi = Integer.MIN\_VALUE;

int s = 0;

for (int i = 0; i < nums.size(); i++) {

s += nums.get(i);

if (s > maxi) {

maxi = s;

}

if (s < 0) {

s = 0;

}

}

System.out.println(maxi);

scanner.close();

}

}



Tc :O(N)

2. 2. Maximum Product Subarray

Given an integer array, the task is to find the maximum product of any subarray.

Input: arr[] = {-2, 6, -3, -10, 0, 2}

Output: 180

Explanation: The subarray with maximum product is {6, -3, -10} with product = 6 \* (-3) \* (-10)

= 180

Input: arr[] = {-1, -3, -10, 0, 60}

Output: 60

Code:

import java.util.\*;

import java.lang.\*;

public class Two {

public static void main(String a[]) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

ArrayList<Integer> nums = new ArrayList<>();

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

nums.add(sc.nextInt());

}

int p1 = nums.get(0);

int p2 = nums.get(0);

int res = nums.get(0);

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

int temp = Math.max(nums.get(i), Math.max(nums.get(i) \* p1, nums.get(i) \* p2));

p2 = Math.min(nums.get(i), Math.min(nums.get(i) \* p1, nums.get(i) \* p2));

p1 = temp;

res = Math.max(p1, res);

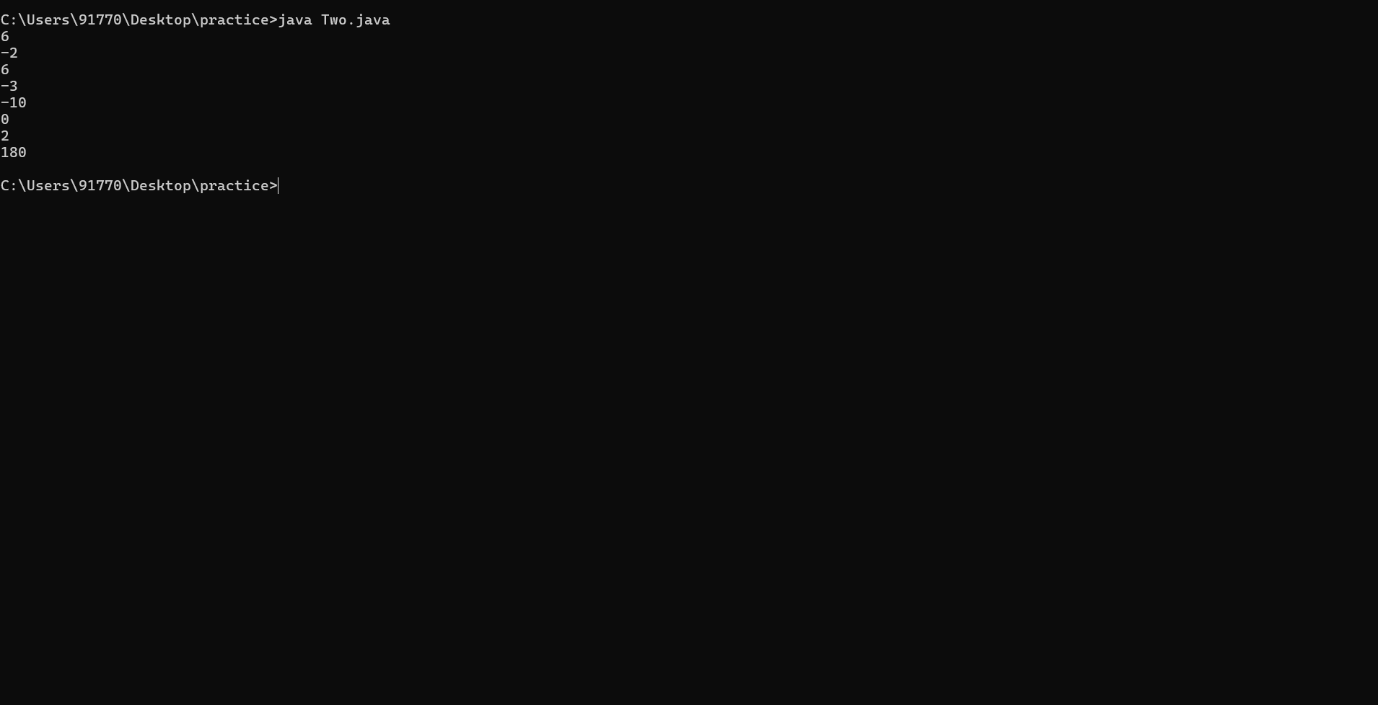
}

System.out.println(res);

}

}

TC:O(N)



3. Search in a sorted and rotated Array

Given a sorted and rotated array arr[] of n distinct elements, the task is to find the index of given

key in the array. If the key is not present in the array, return -1.

Input : arr[] = {4, 5, 6, 7, 0, 1, 2}, key = 0

Output : 4

Input : arr[] = { 4, 5, 6, 7, 0, 1, 2 }, key = 3

Output : -1

Input : arr[] = {50, 10, 20, 30, 40}, key = 10

Output : 1

Code:

import java.util.\*;

import java.lang.\*;

public class Three {

public static void main(String a[]) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

ArrayList<Integer> nums = new ArrayList<>();

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

nums.add(sc.nextInt());

}

int target = sc.nextInt();

int low = 0;

int high = n - 1;

int ans = -1;

while (low <= high) {

int mid = (low + high) / 2;

if (nums.get(mid) == target) {

ans = mid;

break;

} else {

if (nums.get(low) <= nums.get(mid)) {

if (target >= nums.get(low) && target < nums.get(mid)) {

high = mid - 1;

} else {

low = mid + 1;

}

} else {

if (target > nums.get(mid) && target <= nums.get(high)) {

low = mid + 1;

} else {

high = mid - 1;

}

}

}

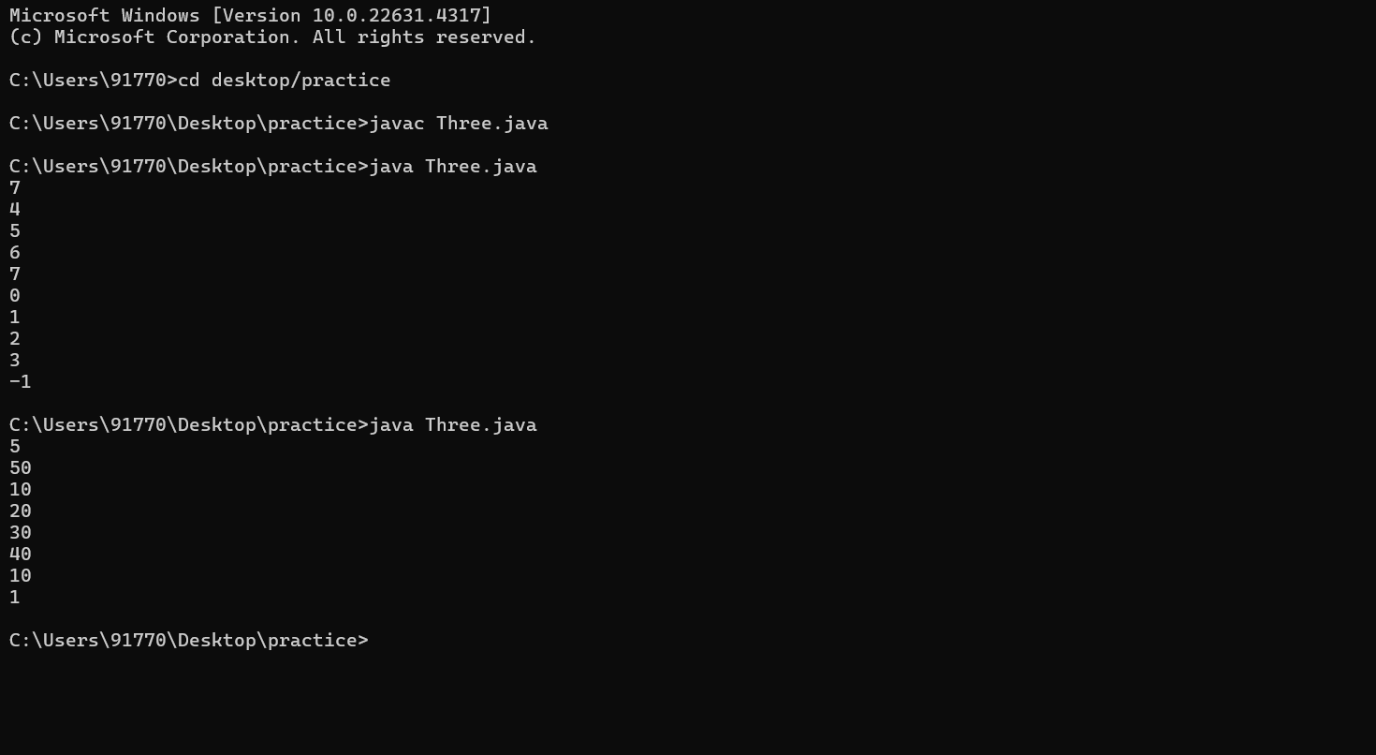
}

System.out.println(ans);

}

}

TC:O(logN)



4.Container with most water

Given n non-negative integers al, ag, ... an where each represents a point at

coordinate (i, an). ` n ' vertical lines are drawn such that the two endpoints of line i

is at (i, a,) and (i, 0). Find two lines, which together with x-axis forms a container,

such that the container contains the most water.

O\D

The program should return an integer which corresponds to the maximum area

of water that can be contained (maximum area instead of maximum volume

sounds weird but this is the 2D plane we are working with for simplicity).

Note: You may not slant the container. Input: arr = [1, 5, 4, 3]

Output: 6

Explanation:

5 and 3 are distance 2 apart. So the size of the base = 2.

Height of container = min(5, 3) = 3. So total area = 3 \* 2 = 6

Input: arr = [3, 1, 2, 4, 5]

Output: 12

Explanation:

5 and 3 are distance 4 apart. So the size of the base = 4.

Height of container = min(5, 3) = 3. So total area = 4 \* 3 = 12

Code:

#include<iostream>

#include<vector>

using namespace std ;

int main(){

vector <int> d;

int n;

cin>>n;

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

int inp;

cin>>inp;

d.push\_back(inp);

}

int l=0;

int r=n-1;

int tank;

int m=INT\_MIN;

while(l<r){

int le=(r-l);

int b=min(d[l],d[r]);

tank=le\*b;

m=max(m,tank);

if(l<r){

l+=1;

}

else{

r-=1;

}

}

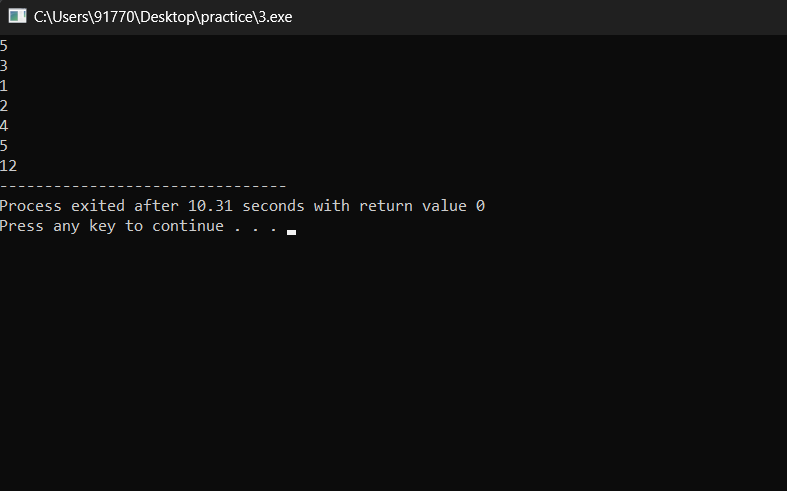
cout<<m;

return 0;

}

TC:O(N)

Output:



5. Find the Factorial of a large number

Input: 100

Output:

933262154439441526816992388562667004907159682643816214685929638952175999932299

156089414639761565182862536979208272237582511852109168640000000000000000000000

00

Input: 50

Output: 30414093201713378043612608166064768844377641568960512000000000000

Code:

import java.util.\*;

import java.math.\*;

public class Five{

    public static void main(String a[]){

        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();

        BigInteger[] dp = new BigInteger[n+1];

        dp[0]=BigInteger.ONE;

        for(int i=1;i<n+1;i++){

            dp[i]=dp[i-1].multiply(BigInteger.valueOf(i));

        }

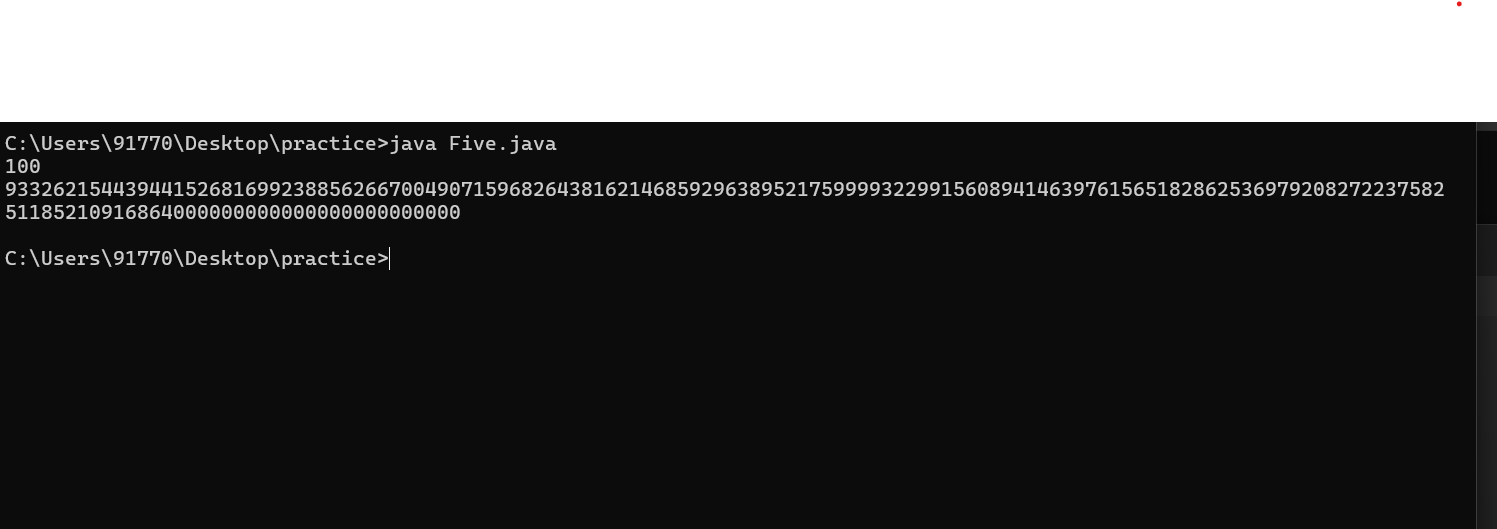
        System.out.println(dp[n]);

    }

}

TC:O(N)

Output:



6. Trapping Rainwater Problem states that given an array of n non-negative integers arr[]

representing an elevation map where the width of each bar is 1, compute how much water it can

trap after rain.

Input: arr[] = {3, 0, 1, 0, 4, 0, 2}

Output: 10

Explanation: The expected rainwater to be trapped is shown in the above image.

Input: arr[] = {3, 0, 2, 0, 4}

Output: 7

Explanation: We trap 0 + 3 + 1 + 3 + 0 = 7 units.

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

Output: 0

Explanation : We cannot trap water as there is no height bound on both sides

Input: arr[] = {10, 9, 0, 5}

Output: 5Code:

#include<iostream>

#include<vector>

using namespace std;

int main(){

vector<int> d;

int n;

cin>>n;

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

int inp;

cin>>inp;

d.push\_back(inp);

}

int total=0;

vector<int> l(n);

vector<int> r(n);

l[0]=d[0];

r[n-1]=d[n-1];

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

l[i]=max(l[i-1],d[i]);

}

for(int i=n-2;i>-1;i--){

r[i]=max(r[i+1],d[i]);

}

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

if(d[i]<l[i] && d[i]<r[i]){

total+=min(r[i],l[i])-d[i];

}

}

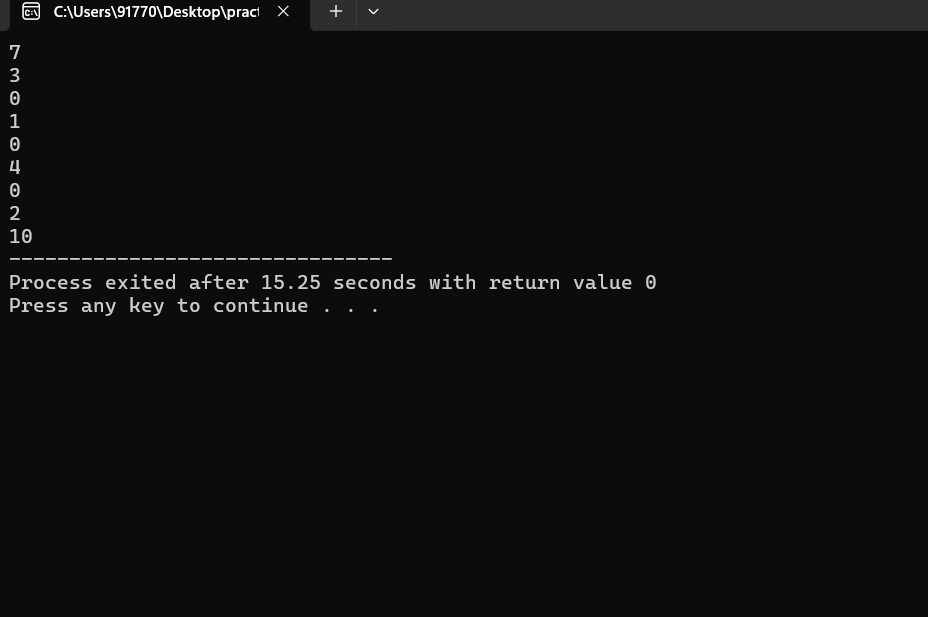
cout<<total;

return 0;

}

TC O(N)

Output:



7. 7. Chocolate Distribution Problem

Given an array arr[] of n integers where arr[i] represents the number of chocolates in ith packet.

Each packet can have a variable number of chocolates. There are m students, the task is to

distribute chocolate packets such that:

Each student gets exactly one packet.

The difference between the maximum and minimum number of chocolates in the packets given

to the students is minimized.

Input: arr[] = {7, 3, 2, 4, 9, 12, 56}, m = 3

Output: 2

Explanation: If we distribute chocolate packets {3, 2, 4}, we will get the minimum difference,

that is 2

Code:

#include<iostream>

#include<vector>

#include<algorithm>

using namespace std;

int main(){

vector <int> d;

int n;

cin>>n;

int m;

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

int inp;

cin>>inp;

d.push\_back(inp);

}

cin>>m;

vector<int> mi;

sort(d.begin(),d.end());

for(int i=0;i<(n-m)+1;i++){

int di=d[i+m-1]-d[i];

mi.push\_back(di);

}

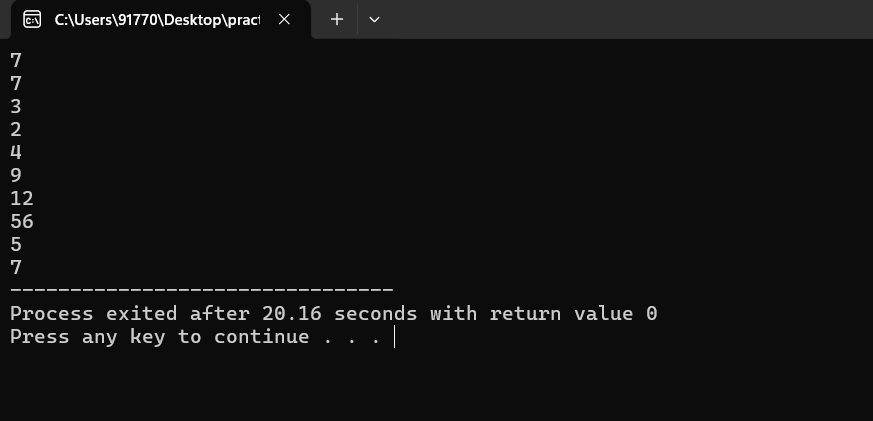
int ans=\*min\_element(mi.begin(),mi.end());

cout<<ans;

return 0;

}

TC O(N)

Output

8. Merge Overlapping Intervals

Given an array of time intervals where arr[i] = [starti, endi], the task is to merge all the

overlapping intervals into one and output the result which should have only mutually exclusive

intervals.

Input: arr[] = [[1, 3], [2, 4], [6, 8], [9, 10]]

Output: [[1, 4], [6, 8], [9, 10]]

Explanation: In the given intervals, we have only two overlapping intervals [1, 3] and [2, 4].

Therefore, we will merge these two and return [[1, 4}], [6, 8], [9, 10]].

Input: arr[] = [[7, 8], [1, 5], [2, 4], [4, 6]]

Output: [[1, 6], [7, 8]]

Explanation: We will merge the overlapping intervals [[1, 5], [2, 4], [4, 6]] into a single interval

[1, 6].

Code:

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

int cols;

cin >> cols;

vector<vector<int>> d(cols, vector<int>(2));

for (int j = 0; j < cols; j++) {

cin >> d[j][0] >> d[j][1];

}

sort(d.begin(), d.end());

vector<vector<int>> c;

for (int i = 0; i < d.size(); i++) {

int start = d[i][0];

int end = d[i][1];

if (!c.empty() && end <= c.back()[1]) {

continue;

}

for (int j = i + 1; j < d.size(); j++) {

if (d[j][0] <= end) {

end = max(end, d[j][1]);

} else {

break;

}

}

c.push\_back({start, end});

}

for (int i = 0; i < c.size(); i++) {

cout << "[" << c[i][0] << ", " << c[i][1] << "] ";

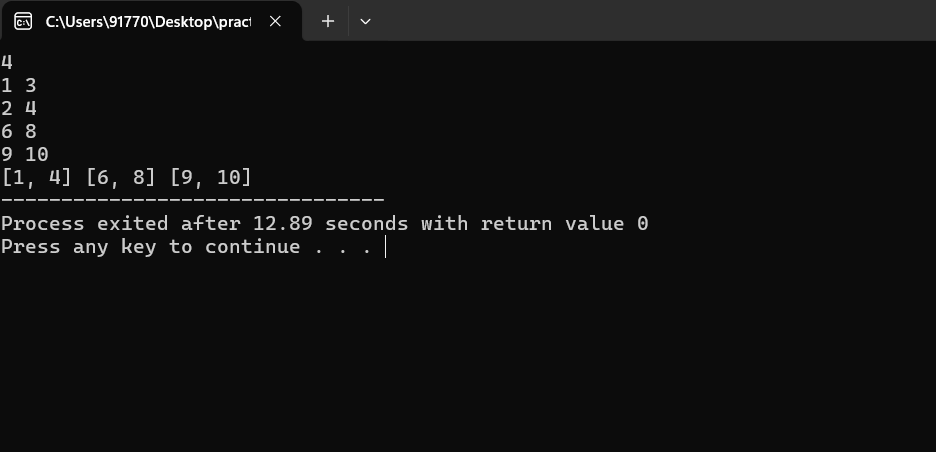
}

return 0;

}

TC:O(NlogN)

Output:



9. A Boolean Matrix Question

Given a boolean matrix mat[M][N] of size M X N, modify it such that if a matrix cell mat[i][j] is

1 (or true) then make all the cells of ith row and jth column as 1.

Input: {{1, 0},

{0, 0}}

Output: {{1, 1}

{1, 0}}

Input: {{0, 0, 0},

{0, 0, 1}}

Output: {{0, 0, 1},

{1, 1, 1}}

Input: {{1, 0, 0, 1},

{0, 0, 1, 0},

{0, 0, 0, 0}}

Output: {{1, 1, 1, 1},

{1, 1, 1, 1},

{1, 0, 1, 1}}

Code:

#include<iostream>

#include<vector>

using namespace std;

int main(){

int rows, cols;

cin >> rows;

cin >> cols;

vector<vector<int>> matrix(rows, vector<int>(cols));

cout << "values" << endl;

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

cin >> matrix[i][j];

}

}

vector<int> ro(rows);

vector<int> co(cols);

for(int i=0;i<rows;i++){

for(int j=0;j<cols;j++){

if(matrix[i][j]==1){

ro[i]=1;

co[j]=1;

}

}

}

for(int i=0;i<rows;i++){

for(int j=0;j<cols;j++){

if(ro[i]==1 || co[j]==1){

matrix[i][j]=1;

}

}

}

for(int i=0;i<rows;i++){

for(int j=0;j<cols;j++){

cout<<matrix[i][j]<<" ";

}

cout<<endl;

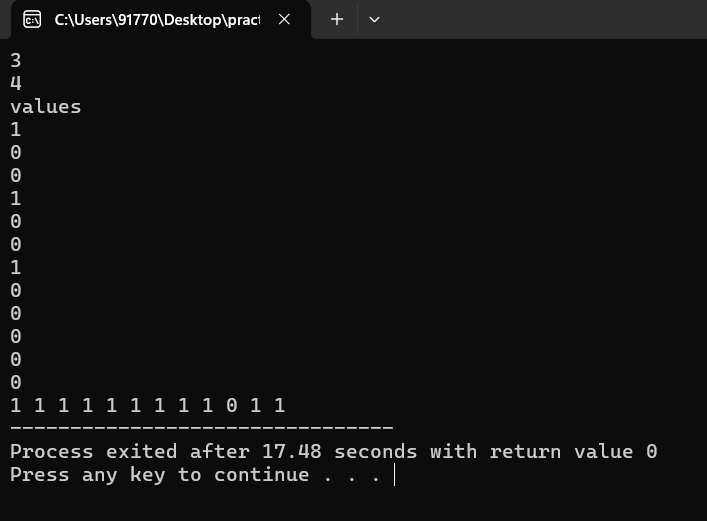
}

return 0;

}

TC O(rc)

Output:



10. Print a given matrix in spiral form

Given an m x n matrix, the task is to print all elements of the matrix in spiral form.

Input: matrix = {{1, 2, 3, 4},

{5, 6, 7, 8},

{9, 10, 11, 12},

{13, 14, 15, 16 }}

Output: 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

Input: matrix = { {1, 2, 3, 4, 5, 6},

{7, 8, 9, 10, 11, 12},

{13, 14, 15, 16, 17, 18}}

Output: 1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11

Explanation: The output is matrix in spiral format.

Code:

#include<iostream>

#include<vector>

using namespace std;

int main(){

int rows, cols;

cin >> rows;

cin >> cols;

vector<vector<int>> matrix(rows, vector<int>(cols));

cout << "values" << endl;

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

cin >> matrix[i][j];

}

}

int top=0;

int bottom=rows-1;

int left=0;

int right = cols-1;

vector<int> ans;

while(top<=bottom && left<=right){

for(int i=left;i<=right;i++){

ans.push\_back(matrix[top][i]);

}

top++;

for(int i=top;i<=bottom;i++){

ans.push\_back(matrix[i][right]);

}

right--;

if(top<=bottom){

for(int i=right;i>=left;i--){

ans.push\_back(matrix[bottom][i]);

}

bottom--;

}

if(left<=right){

for(int i=bottom;i>=top;i--){

ans.push\_back(matrix[i][left]);

}

left++;

}

}

for(int i=0;i<ans.size();i++){

cout<<ans[i]<<" ";

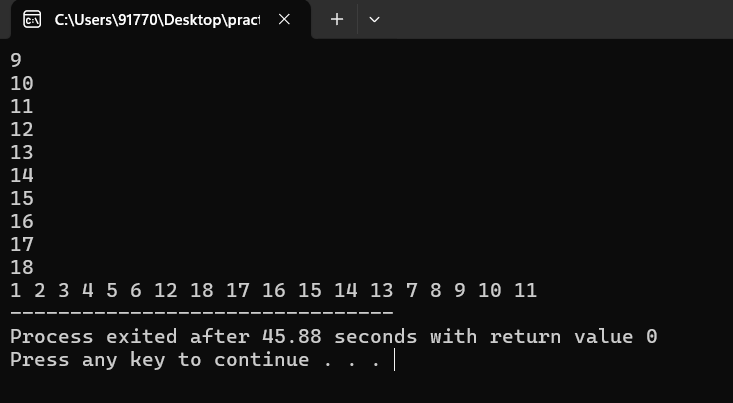
}

return 0;

}

TC:O(rc)

Output:



11.Check if given Parentheses expression is balanced or not

Given a string str of length N, consisting of „(„ and „)„ only, the task is to check whether it is

balanced or not.

Input: str = “((()))()()”

Output: Balanced

Input: str = “())((())”

Output: Not Balanced

Code:

#include<iostream>

#include<string>

#include<stack>

using namespace std;

int main(){

string s;

cin>>s;

stack<char> brackets;

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

if (s[i] == '(') {

brackets.push(s[i]);

} else if (s[i] == ')') {

if (brackets.empty()) {

cout << "Not Balanced" << endl;

}

brackets.pop();

}

}

if (brackets.empty()) {

cout << "Balanced";

} else {

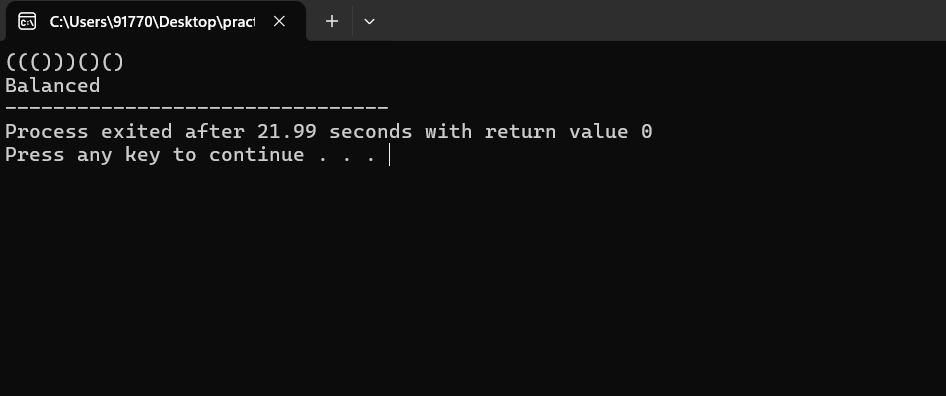
cout << "Not Balanced";

}

}

TC:O(N)

Output:



12.Check if two Strings are Anagrams of each other

Given two strings s1 and s2 consisting of lowercase characters, the task is to check whether the

two given strings are anagrams of each other or not. An anagram of a string is another string that

contains the same characters, only the order of characters can be different.

Input: s1 = “geeks” s2 = “kseeg”

Output: true

Explanation: Both the string have same characters with same frequency. So, they are anagrams.

Input: s1 = “allergy” s2 = “allergic”

Output: false

Explanation: Characters in both the strings are not same. s1 has extra character „y‟ and s2 has

extra characters „i‟ and „c‟, so they are not anagrams.

Input: s1 = “g”, s2 = “g”

Output: true

Explanation: Characters in both the strings are same, so they are anagrams.

Code:

#include <iostream>

using namespace std;

int main(){

string s1;

string s2;

cin>>s1;

cin>>s2;

int c1[26] = {0};

int c2[26] = {0};

string ans=" ";

for (int i = 0; i < s1.size(); i++) {

c1[s1[i] - 'a']++;

c2[s2[i] - 'a']++;

}

for (int i = 0; i < 26; i++) {

if (c1[i] != c2[i]) {

ans="false";

break;

}

}

if(ans!="false"){

ans="true";

}

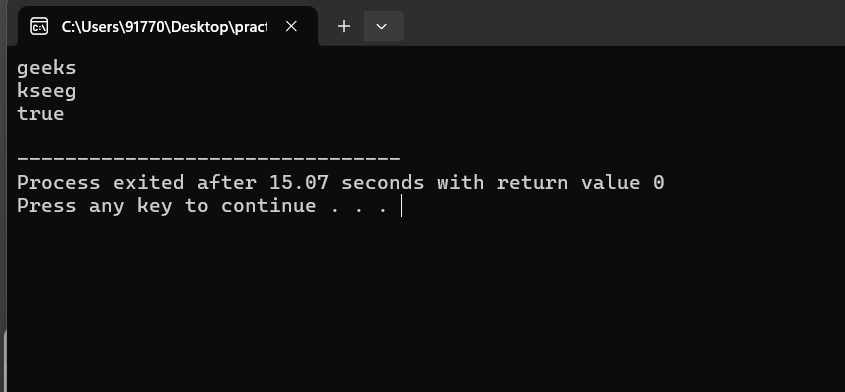
cout << ans << endl;

return 0;

}

TC O(N);

Output:



13. Longest Palindromic Substring

Given a string str, the task is to find the longest substring which is a palindrome. If there are

multiple answers, then return the first appearing substring.

Input: str = “forgeeksskeegfor”

Output: “geeksskeeg”

Explanation: There are several possible palindromic substrings like “kssk”, “ss”, “eeksskee” etc.

But the substring “geeksskeeg” is the longest among all.

Input: str = “Geeks”

Output: “ee”

Input: str = “abc”

Output: “a”

Input: str = “”

Output: “”

Code:

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

int main() {

string s;

cin>>s;

string c = "";

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

for (int j = i + 1; j <= s.size(); j++) {

string sub = s.substr(i, j - i);

string rev = sub;

reverse(rev.begin(), rev.end());

if (sub == rev) {

if (c.size() < sub.size()) {

c = sub;

}

}

}

}

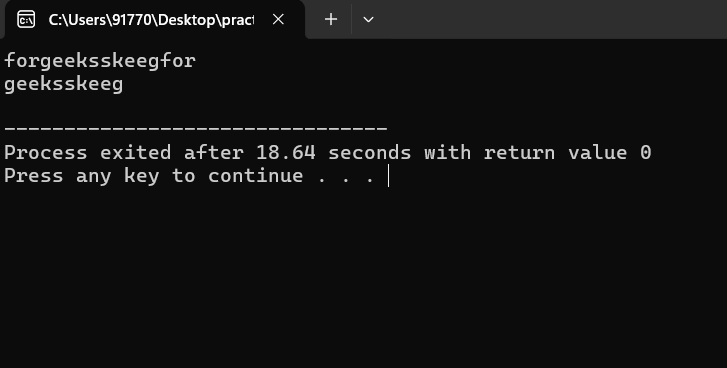
cout << c << endl;

return 0;

}

TC O(n2)

Output:



14. . Longest Common Prefix using Sorting

Given an array of strings arr[]. The task is to return the longest common prefix among each and

every strings present in the array. If there‟s no prefix common in all the strings, return “-1”.

Input: arr[] = [“geeksforgeeks”, “geeks”, “geek”, “geezer”]

Output: gee

Explanation: “gee” is the longest common prefix in all the given strings.

Input: arr[] = [“hello”, “world”]

Output: -1

Explanation: There‟s no common prefix in the given strings.

Code:

#include <iostream>

#include <vector>

#include <string>

using namespace std;

int main() {

int n;

cin >> n;

vector<string> arr(n);

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

cin >> arr[i];

}

string c1 = arr[0];

string a = "";

int l = c1.length();

for (int i = 0; i < l; i++) {

int c = 0;

for (int j = 0; j < n; j++) {

if (c1.substr(0, i + 1) == arr[j].substr(0, i + 1)) {

c++;

}

}

if (c == n) {

a = c1.substr(0, i + 1);

}

}

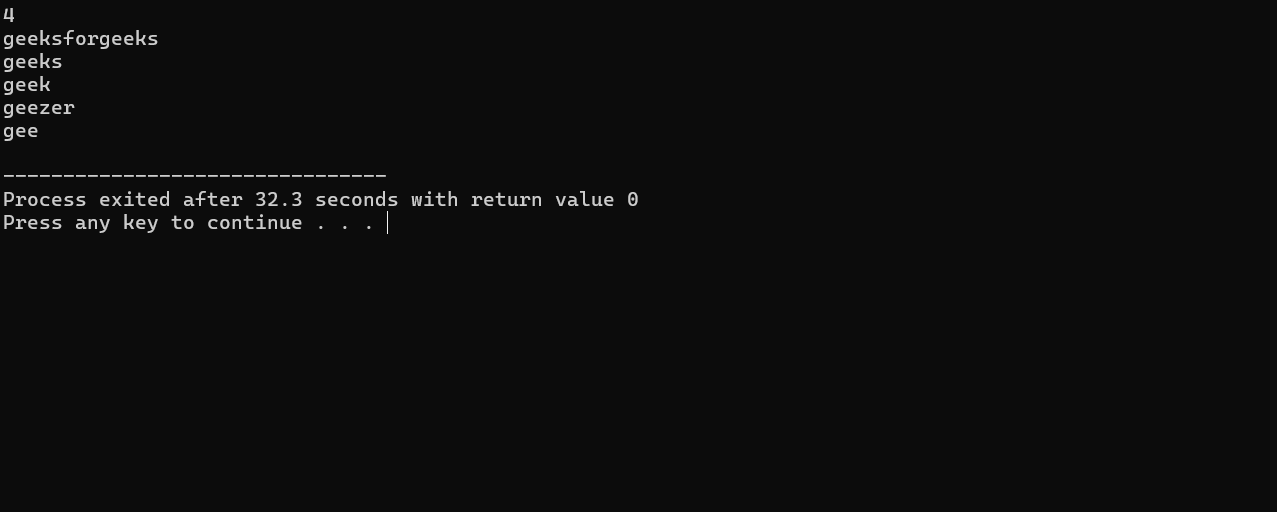
cout << a << endl;

return 0;

}

TC:O(n2)

Output;



15: Delete middle element of a stack

Given a stack with push(), pop(), and empty() operations, The task is to delete the middle element

of it without using any additional data structure.

Input : Stack[] = [1, 2, 3, 4, 5]

Output : Stack[] = [1, 2, 4, 5]

Input : Stack[] = [1, 2, 3, 4, 5, 6]

Output : Stack[] = [1, 2, 4, 5, 6]

Code:

#include <iostream>

#include <stack>

using namespace std;

int main() {

int n;

cin >> n;

stack<int> s;

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

int e;

cin >> e;

s.push(e);

}

int mid = n / 2;

stack<int> s1;

for (int i = 0; i < mid; i++) {

s1.push(s.top());

s.pop();

}

s.pop();

while (!s.empty()) {

s1.push(s.top());

s.pop();

}

while (!s1.empty()) {

cout << s1.top() << " ";

s1.pop();

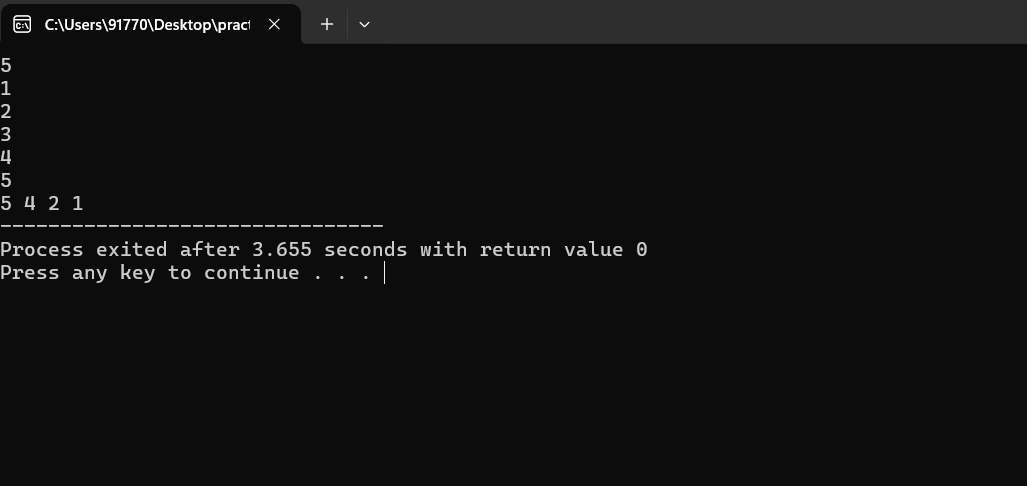
}

return 0;

}

TC O(N)

Output:



16Next Greater Element (NGE) for every element in given Array

Given an array, print the Next Greater Element (NGE) for every element.

Note: The Next greater Element for an element x is the first greater element on the right side of x

in the array. Elements for which no greater element exist, consider the next greater element as -1.

Input: arr[] = [ 4 , 5 , 2 , 25 ]

Output: 4 –> 5

5 –> 25

2 –> 25

25 –> -1

Explanation: Except 25 every element has an element greater than them present on the right side

Input: arr[] = [ 13 , 7, 6 , 12 ]

Output: 13 –> -1

7 –> 12

6 –> 12

12 –> -1

Explanation: 13 and 12 don‟t have any element greater than them present on the right sideCode:

#include <iostream>

#include <stack>

using namespace std;

int main() {

int n;

cin >> n; int nums[n];

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

cin >> nums[i];

}

int ans[n];

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

ans[i] = -1;

}

stack<int> s;

for (int i = 2 \* n - 1; i >= 0; --i) {

while (!s.empty() && s.top() <= nums[i % n]) {

s.pop();

}

if (i < n && !s.empty()) {

ans[i % n] = s.top();

}

s.push(nums[i % n]);

}

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

cout << ans[i] << " ";

}

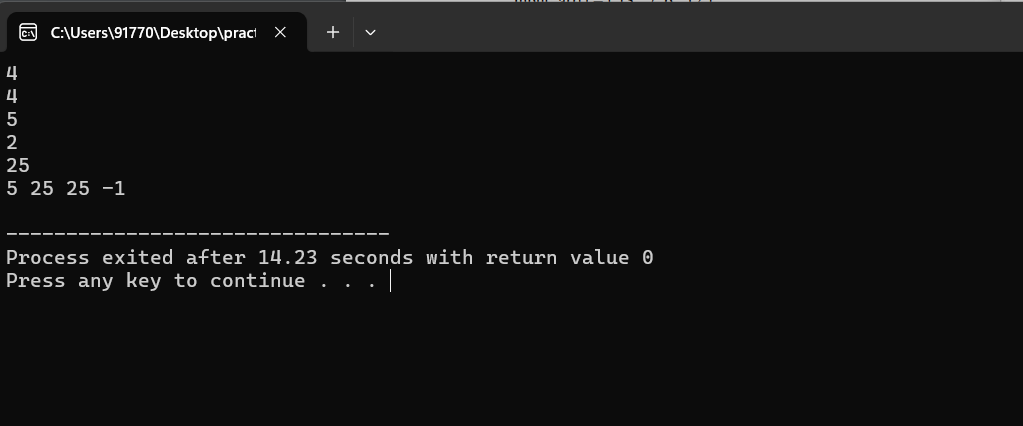
cout << endl;

return 0;

}

TC O(N)

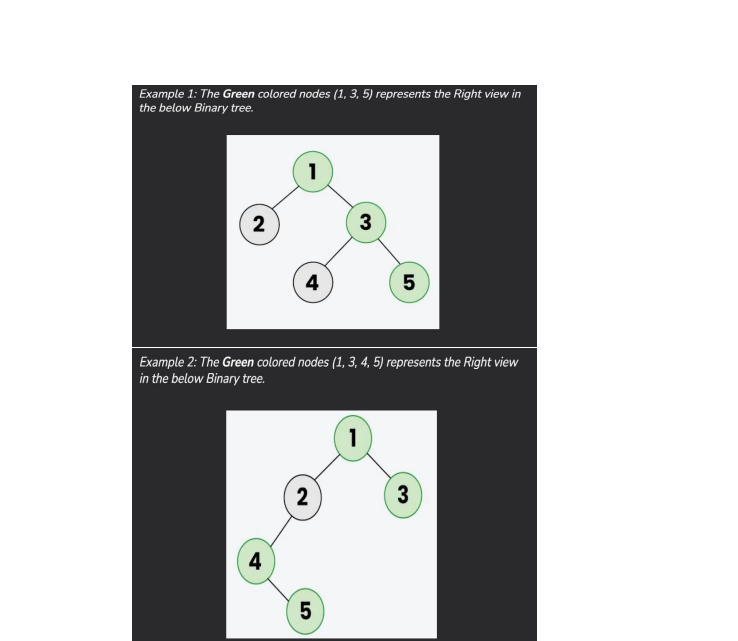
Output:



17: . Print Right View of a Binary Tree

Given a Binary Tree, the task is to print the Right view of it. The right view of a Binary Tree is a

set of rightmost nodes for every level.



Code:

#include <bits/stdc++.h>

using namespace std;

class Node {

public:

int d;

Node \*l, \*r;

Node(int x) {

d = x;

l = r = nullptr;

}

};

void R(Node\* t, int l, int& m, vector<int>& res) {

if (!t) return;

if (l > m) {

res.push\_back(t->d);

m = l;

}

R(t->r, l + 1, m, res);

R(t->l, l + 1, m, res);

}

vector<int> rightView(Node \*t) {

vector<int> res;

int m = -1;

R(t, 0, m, res);

return res;

}

void print(vector<int>& arr) {

for (int v : arr) {

cout << v << " ";

}

cout << endl;

}

int main() {

Node \*t = new Node(1);

t->l = new Node(2);

t->r = new Node(3);

t->r->l = new Node(4);

t->r->r = new Node(5);

vector<int> res = rightView(t);

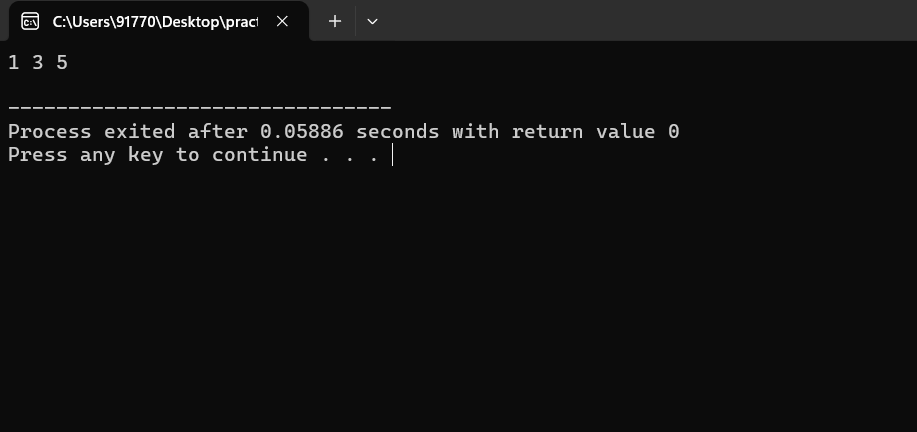
print(res);

return 0;

}

TC:O(N)

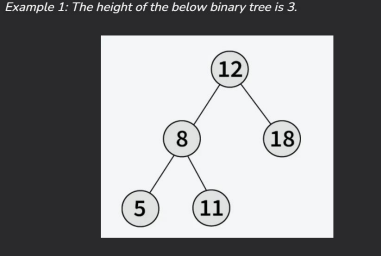
Output:

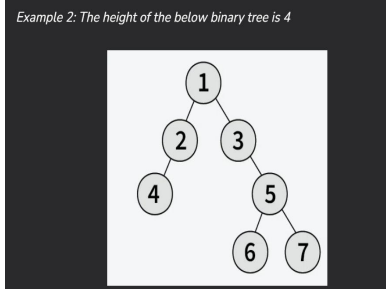


18: Maximum Depth or Height of Binary Tree

Given a binary tree, the task is to find the maximum depth or height of the tree. The height of the

tree is the number of vertices in the tree from the root to the deepest node.





Code:

#include <bits/stdc++.h>

using namespace std;

struct Node {

int data;

Node \*left;

Node \*right;

Node(int val) {

data = val;

left = nullptr;

right = nullptr;

}

};

int maxDepth(Node \*node) {

if (node == nullptr)

return 0;

int lDepth = maxDepth(node->left);

int rDepth = maxDepth(node->right);

return max(lDepth, rDepth) + 1;

}

int main() {

Node \*root = new Node(1);

root->left = new Node(2);

root->right = new Node(3);

root->left->left = new Node(4);

root->left->right = new Node(5);

cout << maxDepth(root);

return 0;

}

TC:O(N)

Output:

