**Coding practice Problems:**

**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.

Input: arr[] = {5, 4, 1, 7, 8}

Output: 25

Explanation: The subarray {5, 4, 1, 7, 8} has the largest sum 25.

**Code:**

import java.util.\*;

import java.io.\*;

public class MaxSum

{

public static void main(String[] siva)

{

Scanner s=new Scanner(System.in);

int n=s.nextInt();

int[] nums=new int[n];

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

{

nums[i]=s.nextInt();

}

if(n==0){

System.out.println(0);

return;

}

if(n==1){

System.out.println(nums[0]);

return;

}

int sofar=Integer.MIN\_VALUE;

int ends=0;

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

{

ends+=nums[i];

if(sofar<ends)

{

sofar=ends;

}

if(ends<0)

{

ends=0;

}

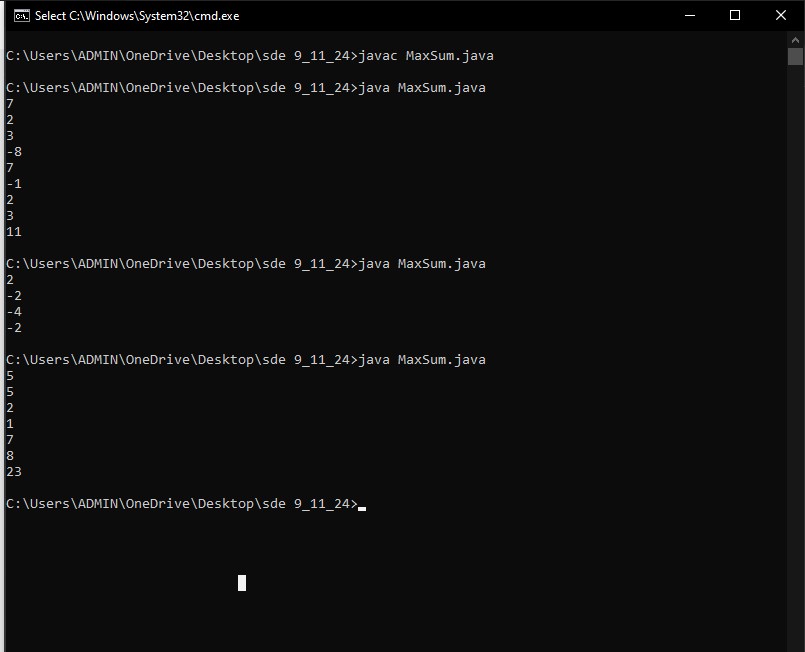
}

System.out.println(sofar);

}

}

**Output:**



Time Complexity : O(n)

**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

Explanation: The subarray with maximum product is {60}.

**Code:**

import java.util.\*;

import java.io.\*;

import java.lang.\*;

class MaxProd{

public static void main(String[] siva)

{

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int[] nums=new int[n];

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

{

nums[i]=sc.nextInt();

}

int ans=nums[0];

int maxi=nums[0];

int mini=nums[0];

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

{

int temp=maxi;

maxi=Math.max(nums[i],Math.max(maxi\*nums[i],mini\*nums[i]));

mini=Math.min(nums[i],Math.min(temp\*nums[i],mini\*nums[i]));

ans=Math.max(ans,maxi);

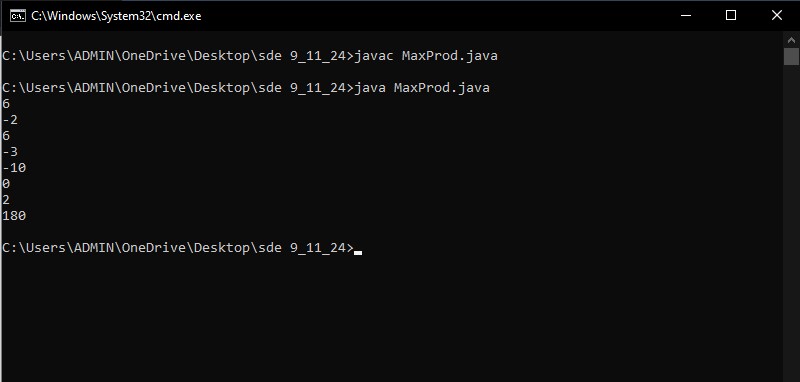
}

System.out.println(ans);

}

}

**Code:**

****

**Time Complexity:** O(N)

**3. Search in a sorted and rotated Array – Binary Search**

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

**Code:**

import java.util.\*;

class Search {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int target = sc.nextInt();

int[] nums = new int[n];

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

nums[i] = sc.nextInt();

}

int low = 0;

int high = n - 1;

int output = -1;

while (low <= high) {

int mid = (low + high) / 2;

if (nums[mid] == target)

{

output = mid;

break;

}

if (nums[low] <= nums[mid]) {

if (nums[low] <= target && target < nums[mid]) {

high = mid - 1;

} else {

low = mid + 1;

}

} else {

if (nums[mid] < target && target <= nums[high]) {

low = mid + 1;

} else {

high = mid - 1;

}

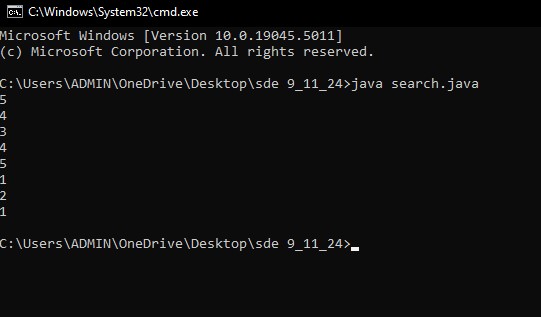
}

}

System.out.println(output);

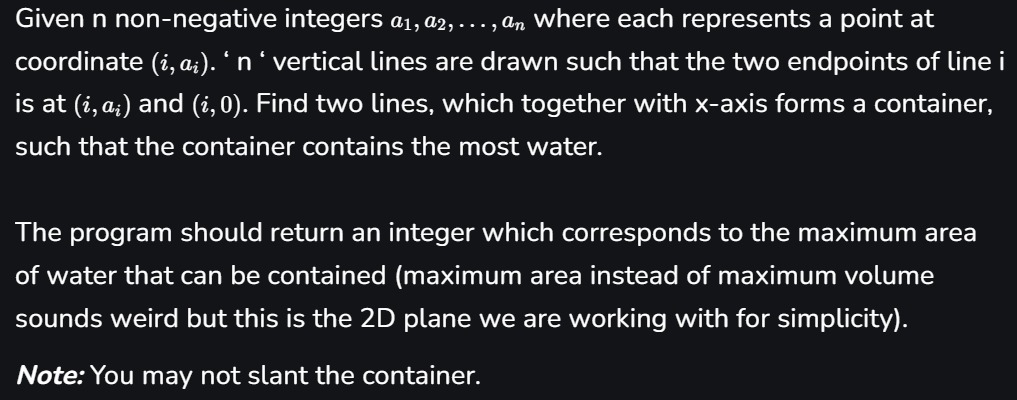
}

}



Time Complexity: O(log N)

**4.Container with most water**

****

**Code:**

#include<iostream>

#include<vector>

using namespace std;

int main()

{

int n;

cout<<"Enter n:";

cin>>n;

vector<int> height;

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

{

int x;

cin>>x;

height.push\_back(x);

}

int left=0;

int right=height.size()-1;

int maxi=0;

while(left<right)

{

int len=(right-left);

int bre=min(height[left],height[right]);

int area=len\*bre;

maxi=max(maxi,area);

if(height[left]<height[right])

{

left++;

}

else

{

right--;

}

}

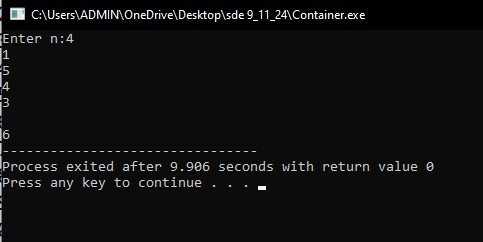
cout<<endl;

cout<<maxi;

return 0;

}

Output:



Time Complexity: O(N)

5. Find the Factorial of a large number

Code:

import java.util.\*;

import java.math.BigInteger;

class largefact{

public static void main(String[] siva){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

BigInteger res=BigInteger.ONE;

for(int i=2;i<n;i++)

{

res=res.multiply(BigInteger.valueOf(i));

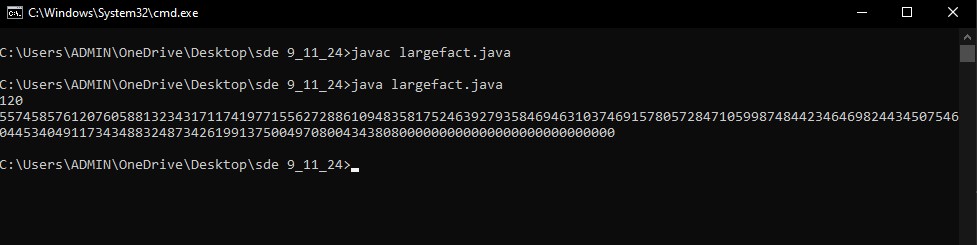
}

System.out.println(res);

}

}

Output:



Time Complexity: O(N)

6.Trapping Rain Water

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

Code:

#include<iostream>

#include<vector>

#include<algorithm>

using namespace std;

int main()

{

int n;

cout<<"Enter n:";

cin>>n;

vector<int> height;

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

{

int x;

cin>>x;

height.push\_back(x);

}

int left = 0;

int right = height.size() - 1;

int leftMax = height[left];

int rightMax = height[right];

int water = 0;

while (left < right) {

if (leftMax < rightMax) {

left++;

leftMax = max(leftMax, height[left]);

water += leftMax - height[left];

} else {

right--;

rightMax = max(rightMax, height[right]);

water += rightMax - height[right];

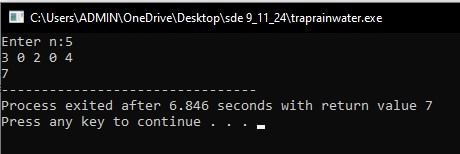
}

}

cout<<water;

}

Output:



Time Complexity : O(N)

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> arr;

int m;

int n;

cout<<"n:";

cin>>n;

cout<<"Enter the values"<<endl;

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

{

int x;

cin>>x;

arr.push\_back(x);

}

cout<<"m:";

cin>>m;

int mini=INT\_MAX;

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

int minr=arr[m-1]-arr[0];

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

{

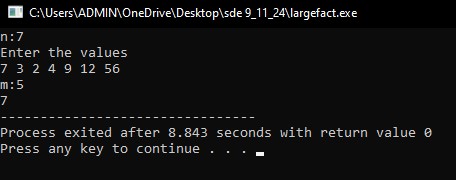
minr=min(minr,arr[i+m]-arr[i]);

}

cout<<minr;

}

Output:



Time Complexity: O(N log N)

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]].

Program:

#include<iostream>

#include<vector>

#include<algorithm>

using namespace std;

int main()

{

int n;

cout<<"enter n:";

cin>>n;

vector<vector<int>> intervals;

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

{

vector<int> a;

for(int j=0;j<2;j++)

{

int temp;

cin>>temp;

a.push\_back(temp);

}

intervals.push\_back(a);

}

sort(intervals.begin(), intervals.end(), [](const vector<int>& a, const vector<int>& b) {

return a[0] < b[0];

});

vector<vector<int>> merged;

vector<int> prev = intervals[0];

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

vector<int> interval = intervals[i];

if (interval[0] <= prev[1]) {

prev[1] = max(prev[1], interval[1]);

} else {

merged.push\_back(prev);

prev = interval;

}

}

merged.push\_back(prev);

cout<<"Output:"<<endl;

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

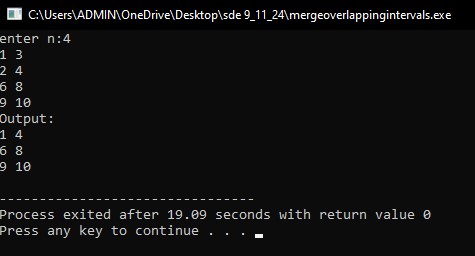
{

cout<<merged[i][0]<<" "<<merged[i][1]<<endl;

}

return 0;

}  
Output:



Time Complexity:O(N log N)

**10.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}}

**Program:**

#include <iostream>

#include <vector>

using namespace std;

int main() {

int M, N;

cout << "Enter number of rows: ";

cin >> M;

cout << "Enter number of columns: ";

cin >> N;

vector<vector<int>> mat(M, vector<int>(N));

cout << "Enter the matrix elements (0 or 1):" << endl;

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

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

cin >> mat[i][j];

}

}

vector<int> row(M, 0);

vector<int> col(N, 0);

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

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

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

row[i] = 1;

col[j] = 1;

}

}

}

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

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

if (row[i] == 1 || col[j] == 1) {

mat[i][j] = 1;

}

}

}

cout << "Modified Matrix:" << endl;

for (const auto& row : mat) {

for (int cell : row) {

cout << cell << " ";

}

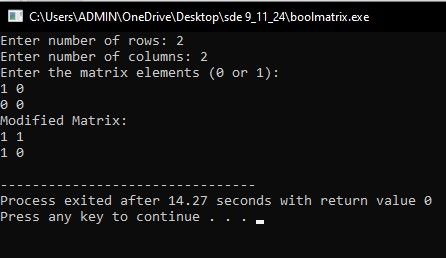
cout << endl;

}

return 0;

}

Output:



Time Complexity: O(M\*N)

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}}

**Program:**

#include<iostream>

#include<vector>

#include<algorithm>

using namespace std;

int main(){

int n,m;

cin>>n;

cin>>m;

vector<vector<int>>matrix(m,vector<int>(n));

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

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

cin >> matrix[i][j];

}

}

vector<int> res(m \* n);

int j = 0;

int l = 0, u = 0, d = m - 1, r = n - 1;

while (l <= r && u <= d) {

if (l <= r && u <= d) {

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

res[j++] = matrix[u][i];

}

u++;

}

if (l <= r && u <= d) {

for (int i = u; i <= d; i++) {

res[j++] = matrix[i][r];

}

r--;

}

if (l <= r && u <= d) {

for (int i = r; i >= l; i--) {

res[j++] = matrix[d][i];

}

d--;

}

if (l <= r && u <= d) {

for (int i = d; i >= u; i--) {

res[j++] = matrix[i][l];

}

l++;

}

}

for (int i=0;i<n\*m;i++)

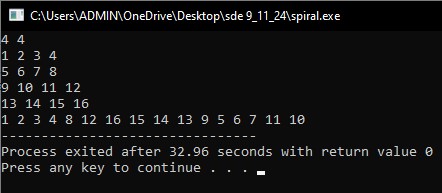
{

cout<<res[i]<<" ";

}

}

Output:



Time Complexity: O(n\*m)

13. 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

**Program:**

#include <iostream>

#include <vector>

#include <string>

using namespace std;

int main() {

string s;

bool f = false;

cin >> s;

int n = s.size();

vector<char> su;

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

{

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

{

su.push\_back('(');

}

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

{

if (!su.empty())

{

su.pop\_back();

}

else

{

f = true;

break;

}

}

}

if (!su.empty() || f)

{

cout << "Not Balanced" << endl;

} else

{

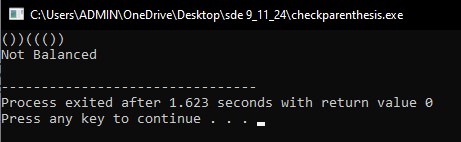
cout << "Balanced" << endl;

}

return 0;

}

**Output:**

****

**Time Complexity:** O(N)

14. 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.

**Program:**

#include <iostream>

#include <vector>

#include <string>

#include<algorithm>

using namespace std;

int main()

{

string a,b;

cin>>a;

cin>>b;

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

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

if(a==b)

{

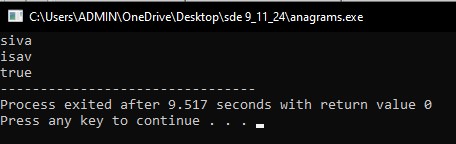
cout<<"true";

}

else cout<<"false";

}

**Output:**

****

**Time Complexity:** O(N log N)

**15. 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.

**Program:**

#include <iostream>

#include <vector>

#include <string>

#include<algorithm>

using namespace std;

int main()

{

string ans;string s;

cin>>s;

for(int i=0;i<2\*s.length()-1;i++){

int left=i/2;

int right=left+i%2;

while(left>=0 && right<s.length() && s[left]==s[right]){

left--;

right++;

}

if(right-left-1>ans.length()){

ans=s.substr(left+1,right-left-1);

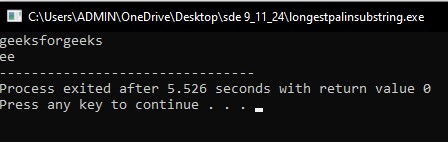
}

}

cout<<ans;

}

**Output:**

****

**Time Complexity:** O(N2)

16. 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.

**Program:**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

int n;

cin >> n;

vector<string> arr(n);

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

cin >> arr[i];

}

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

string first = arr[0];

string last = arr[arr.size() - 1];

int i = 0;

while (i < first.length() && i < last.length() && first[i] == last[i]) {

i++;

}

if (i == 0) {

cout << "-1" << endl;

} else {

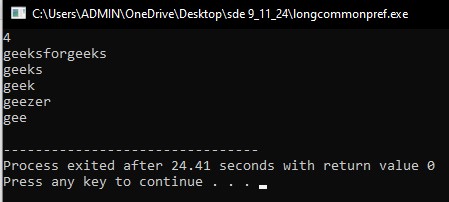
cout << first.substr(0, i) << endl;

}

return 0;

}

Output:



Time Complexity:O(nlogn⋅m)

17. 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]

**Program:**

#include <iostream>

#include <stack>

using namespace std;

int main() {

int n;

cin >> n;

stack<int> s;

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

int x;

cin >> x;

s.push(x);

}

int mid = n / 2;

stack<int> temp;

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

temp.push(s.top());

s.pop();

}

s.pop();

while (!s.empty()) {

temp.push(s.top());

s.pop();

}

while (!temp.empty()) {

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

temp.pop();

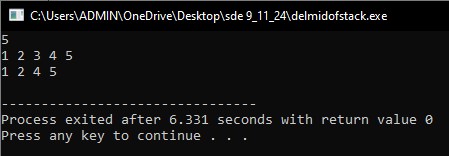
}

cout << endl;

return 0;

}

Output:



Time Complexity: O(N)

18. Next 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

**Program:**

#include <iostream>

#include <stack>

#include <vector>

using namespace std;

int main() {

int n;

cin >> n;

vector<int> arr(n);

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

cin >> arr[i];

}

stack<int> s;

vector<int> nge(n, -1);

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

while (!s.empty() && arr[s.top()] < arr[i]) {

nge[s.top()] = arr[i];

s.pop();

}

s.push(i);

}

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

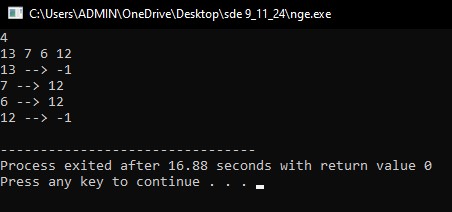
cout << arr[i] << " --> " << nge[i] << endl;

}

return 0;

}

Output:



Time Complexity: O(N)