

**Ahsania Mission University of Science & Technology**

**Lab Report**

**Lab No:** 03

**Course Code:** CSE2202

**Course Title:** Computer Algorithm Sessional.

**Submitted By:**

Afrin Nahrin Nipu

ID: 1012320005101025

1st Batch, 2nd Year, 2nd Semester

Department of Computer science and Engineering,

Ahsania Mission University of Science & Technology

**Submitted To:**

Md. Fahim Faisal

Lecturer,

Department of Computer science and Engineering,

Ahsania Mission University of Science & Technology

**Task No.:** 01

**Problem Statement:** Maximum Subarray Sum

* Given an integer array numsnums, find the subarray with the largest sum, and print its sum.
* Note: A subarray is a contiguous non-empty sequence of elements within an array.

Input Format

* The first line contains T, the number of test cases.
* The first line in each test case contains N, the number of elements in an array.
* The second line in each test case contains N integers, denoting the elements in the array.

Output Format

For each test case, output the maximum subarray sum of each array.

Constraints

• 1≤ T≤100

• 1≤ N≤100

• −109≤ Ai≤109

Input

3

9

-2 1 -3 4 -1 2 1 -5 4

1

1

5

5 4 -1 7 8

Output

6

1

23

**Source Code:**

#include <bits/stdc++.h>

using namespace std;

int main()

{

int t;

cin >> t;

while(t--)

{

int n;

cin >> n;

int arr[n];

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

{

cin >> arr[i];

}

int maxsum = INT\_MIN;

// Iterate over all possible subarrays

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

{

int currentSum = 0;

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

{

currentSum += arr[j]; // Add current element to the sum

if(currentSum>maxsum)

{

maxsum = currentSum; // Update maxsum if the currentSum is larger

}

}

}

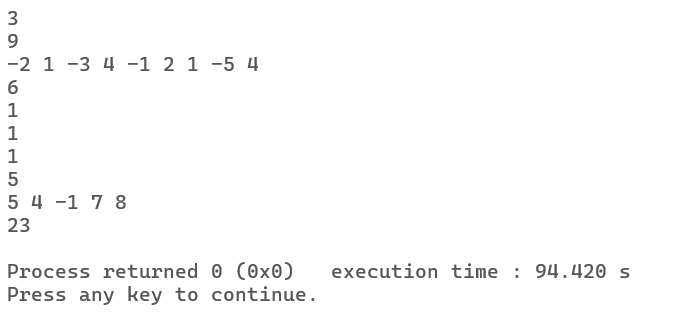
cout << maxsum << endl;

}

return 0;

}

**Output:**

****

**Task No.:** 02

**Problem Statement** : Implement Insertion Sort Algorithm.

**Source Code:**

#include <iostream>

using namespace std;

void insertionSort(int arr[], int n)

{

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

{

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key)

{

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

void printArray(int arr[], int n)

{

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

{

cout << arr[i] << " ";

}

cout << endl;

}

int main()

{

int arr[] = {8, 2, 4, 9, 3, 6};

int n = sizeof(arr) / sizeof(arr[0]);

cout << "Original Array: ";

printArray(arr, n);

insertionSort(arr, n);

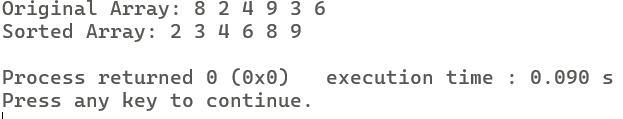
cout << "Sorted Array: ";

printArray(arr, n);

return 0;

}

**Output:**

****

**Task No.:** 03

**Problem Statement:** Implement Merge Sort algorithm.

**Source Code:**

#include <iostream>

using namespace std;

void merge(int arr[], int left, int mid, int right)

{

int n1 = mid - left + 1;

int n2 = right - mid;

int L[n1], R[n2];

for (int i = 0; i < n1; i++)

L[i] = arr[left + i];

for (int j = 0; j < n2; j++)

R[j] = arr[mid + 1 + j];

int i = 0, j = 0, k = left;

while (i < n1 && j < n2)

{

if (L[i] <= R[j])

{

arr[k] = L[i];

i++;

}

else

{

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1)

{

arr[k] = L[i];

i++;

k++;

}

while (j < n2)

{

arr[k] = R[j];

j++;

k++;

}

}

void mergeSort(int arr[], int left, int right)

{

if (left < right)

{

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

mergeSort(arr, left, mid);

mergeSort(arr, mid + 1, right);

merge(arr, left, mid, right);

}

}

void printArray(int arr[], int n)

{

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

{

cout << arr[i] << " ";

}

cout << endl;

}

int main()

{

int n;

cin>>n;

int arr[n];

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

{

cin>> arr[i];

}

cout << "Original Array: ";

printArray(arr, n);

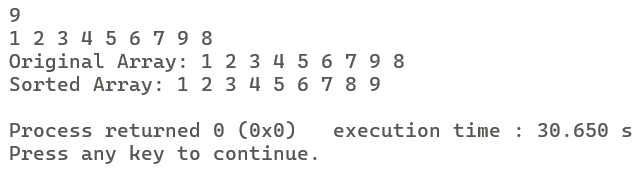
mergeSort(arr, 0, n - 1);

cout << "Sorted Array: ";

printArray(arr, n);

return 0;

}

**Output: **

**Task No.:** 04

**Problem Statemen:** Countiversion using merge sort algorithm.

**Source Code:**

#include <iostream>

using namespace std;

int countAndMerge(int arr[], int l, int m, int r) {

int n1 = m - l + 1;

int n2 = r - m;

int left[n1], right[n2];

for (int i = 0; i < n1; i++) {

left[i] = arr[l + i];

}

for (int i = 0; i < n2; i++) {

right[i] = arr[m + 1 + i];

}

int res = 0, i = 0, j = 0, k = l;

while (i < n1 && j < n2) {

if (left[i] <= right[j]) {

arr[k] = left[i];

i++;

} else {

arr[k] = right[j];

j++;

res += (n1 - i);

}

k++;

}

while (i < n1) {

arr[k] = left[i];

i++;

k++;

}

while (j < n2) {

arr[k] = right[j];

j++;

k++;

}

return res;

}

int countInv(int arr[], int l, int r) {

int res = 0;

if (l < r) {

int m = l + (r - l) / 2;

res = res + countInv(arr, l, m);

res = res + countInv(arr, m + 1, r);

res = res + countAndMerge(arr, l, m, r);

}

return res;

}

int main() {

int n;

cin>>n;

int arr[n];

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

{

cin>> arr[i];

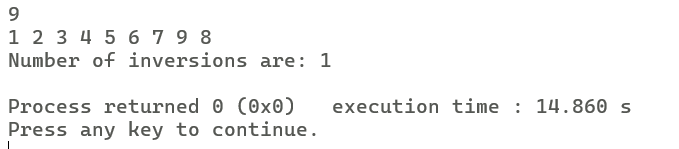
}

cout << "Number of inversions are: " << countInv(arr, 0, n - 1) << endl;

return 0;

}

**Output:**

****

**Task No.:** 05

**Problem Statement:** Implement linier search algorithm.

**Source Code:**

#include <iostream>

using namespace std;

int main()

{

int n;

cin>>n;

int arr[n];

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

{

cin>>arr[i];

}

int s;

cout<<"Enter the searching number: ";

cin>>s;

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

{

if(s == arr[i])

{

cout<<"INDEX: ";

cout<<i<<endl;

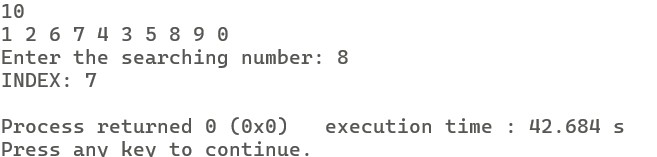
break;

}

}

return 0;

}

**Output: **

**Task No.:** 06

**Problem Statement:** Problem: Leader of an Array

Write a program to print all the leaders in the array. An element is a leader if it is strictly greater than all the elements to its right side. And the rightmost element is always a leader.

Input Format

* + The first line contains N, the number of elements in an array.
  + The second line contains N integers, denoting the elements in the array.

Output Format

In a single line output all the leaders in the given array.

Sample 1:

Input

6

16 17 4 3 5 2

Output

17 5 2

**Source Code:**

#include <iostream>

using namespace std;

int main()

{

int n;

cin >> n;

int arr[1000];

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

{

cin >> arr[i];

}

int max\_from\_right = arr[n - 1];

cout << max\_from\_right << " ";

for (int i = n - 2; i >= 0; i--)

{

if (arr[i] > max\_from\_right)

{

max\_from\_right = arr[i];

cout << max\_from\_right << " ";

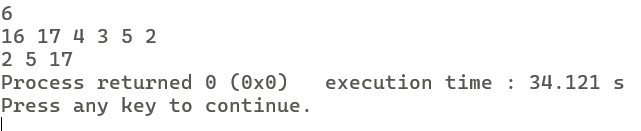
}

}

return 0;

}

**Output:**

****