**DS LAB WEEK 3A – F1**

**9923103011 – MOHD TARIQ SAMI**

**Q1. Find the first pair of repeating elements in a given array by either returning its**

**position(index) or a suitable statement if the set is not found or not repeated.**

**Assumption 1: indexing starts with value 1.**

**Assumption 2: There are ‘n’ number of elements in the array and you should allow**

**for it to be dynamically decided .**

**Example:**

**Input: A= {3,6,12,-10,3,3,6,34,0, -109,98,1}**

**Find the pair: 3,6**

**Output:6**

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

int pairfind(int n,int arr[])

{

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

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

if (arr[i - 1] == arr[j - 1] && arr[i] == arr[j]) {

return i;

}

}

}

return -1;

}

int main()

{

int n;

cout<<"Enter total array elements: ";

cin>>n;

int\* arr=new int[n];

cout<<"Enter array elements: ";

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

{

cout<<"Enter the "<<i+1<<" element: ";

cin>>arr[i];

}

int ans=pairfind(n,arr);

if(ans!=-1)

{

cout<<"The position of pair is: "<<ans<<endl;

}

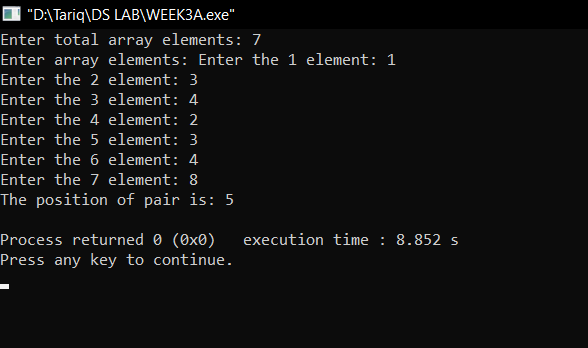
else

{

cout<<"The pair does not repeat"<<endl;

}

}

****

**Q2.Given an integer array of ‘N’ elements. You need to find the maximum sum of**

**two elements such that the sum is closest to a given value, say zero (try with**

**visualizing the number line).**

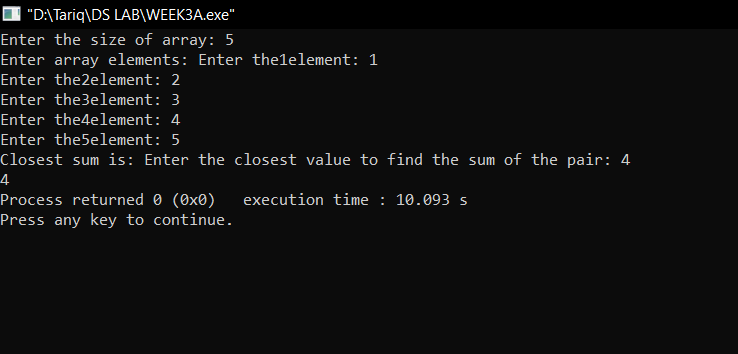
**Example:**

**Input: N = 3, arr[] = {-5, -50, 56}**

**Output: -68**

**Explanation: Sum of two elements closest to zero is 56.**

**-55 + (-5) = - 55 -5+56 = 51 -50 +56 =6**

****

int closestsum(int n,vector<int> &arr)

{

int low=0;

int high=n-1;

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

int closest,sum;

cout<<"Enter the closest value to find the sum of the pair: ";

cin>>closest;

while(low<high)

{

int sum=arr[low]+arr[high];

if(sum<closest)

{

low++;

}

else if(sum>closest)

{

high--;

}

else

{

return sum;

}

}

return sum;

}

int main()

{

int n;

cout<<"Enter the size of array: ";

cin>>n;

vector<int> arr;

int ele;

cout<<"Enter array elements: ";

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

{

cout<<"Enter the"<<i+1<<"element: ";

cin>>ele;

arr.push\_back(ele);

}

cout<<"Closest sum is: "<<closestsum(n,arr);

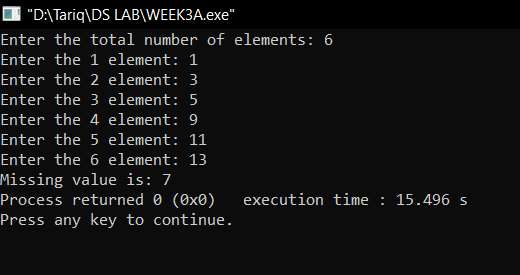
}

**Q3. You are given an array ‘A’, which contains elements arranged in an Arithmetic**

**Progression. Within this pre-sorted set of elements (A.P.) one of the elements is**

**missing. Write a program by making a user-defined function called**

**‘findMissingAPval’ to find this missing element.**

****

int findMissingAPval(const vector<int>& arr) {

int n = arr.size();

int diff = (arr[n - 1] - arr[0]) / n;

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

if (arr[i] - arr[i - 1] != diff) {

return arr[i - 1] + diff;

}

}

}

int main()

{

cout<<"Enter the total number of elements: ";

int n,ele;

cin>>n;

vector<int> arr;

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

{

cout<<"Enter the "<<i+1<<" element: ";

cin>>ele;

arr.push\_back(ele);

}

cout<<"Missing value is: ";

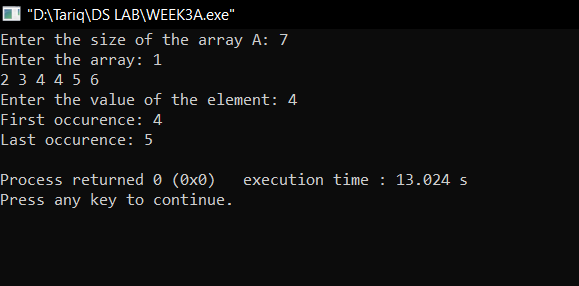
cout<<findMissingAPval(arr);

}

**Q4. You are given a sorted array ‘A’ with possibly duplicate elements. The task is**

**to find indexes(positions) of the first and last occurrences of an element ‘x’ in the**

**given array. Write an efficient program to perform the above requirement.**



void occurence(int arr[],int n)

{

int x;

cout<<"Enter the value of the element: ";

cin>>x;

int first=0,c=0;

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

{

if(arr[i]==x)

{

first=i;

while(arr[i]==arr[i+c])

{

c++;

}

}

}

cout<<"First occurence: "<<first<<endl<<"Last occurence: "<<first+c-1<<endl;

}

int main()

{

cout<<"Enter the size of the array A: ";

int n;

cin>>n;

int \* arr=new int[n];

cout<<"Enter the array: ";

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

{

cin>>arr[i];

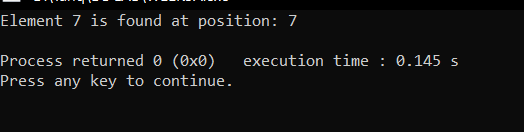
}

occurence(arr,n);

}

**Q5. Given a sorted array of size N and an integer K, find the position at which K is**

**present in the array using interpolation search.**

****

int interpolationSearch(const vector<int>& arr, int K) {

int low = 0;

int high = arr.size() - 1;

while (low <= high && K >= arr[low] && K <= arr[high]) {

int pos = low + ((K - arr[low]) \* (high - low)) / (arr[high] - arr[low]);

if (arr[pos] == K) {

return pos + 1;

}

if (arr[pos] < K) {

low = pos + 1;

}

else {

high = pos - 1;

}

}

return -1;

}

int main() {

vector<int> arr = {1,2,3,4,5,6,7,5,3,32,22};

int K = 7;

int position = interpolationSearch(arr, K);

if (position != -1) {

cout << "Element " << K << " is found at position: " << position << endl;

} else {

cout << "Element " << K << " is not found in the array." << endl;

}

return 0;

}

**Q6. Use Binary search to find the maximum length subarray such that its first**

**element is greater than or equal to the last element of the subarray in a given**

**array arr[0..n-1] of length ‘n’.**

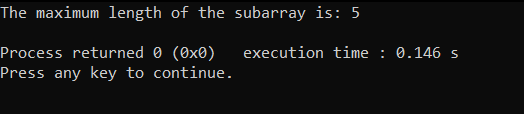
**Example:**

**Input : arr[] = {-5, -1, 7, 5, 1, -2}**

**Output : 5**

**Hint: Subarray {-1, 7, 5, 1, -2} forms a maximum length subarray with its**

**first element greater than last.**

****

bool isValidSubarray(const vector<int>& arr, int length) {

int n = arr.size();

// Iterate over all possible starting points of the subarray of the given length

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

if (arr[i] >= arr[i + length - 1]) {

return true;

}

}

return false;

}

int findMaxLengthSubarray(const vector<int>& arr) {

int n = arr.size();

int left = 1, right = n;

int maxLength = 0;

while (left <= right) {

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

if (isValidSubarray(arr, mid)) {

maxLength = mid;

left = mid + 1;

} else {

right = mid - 1;

}

}

return maxLength;

}

int main() {

vector<int> arr = {-5, -1, 7, 5, 1, -2};

int result = findMaxLengthSubarray(arr);

cout << "The maximum length of the subarray is: " << result << endl;

return 0;

}

**Q7. Given an array of Strings and a String x, find an index of x if it is present in the**

**array.**

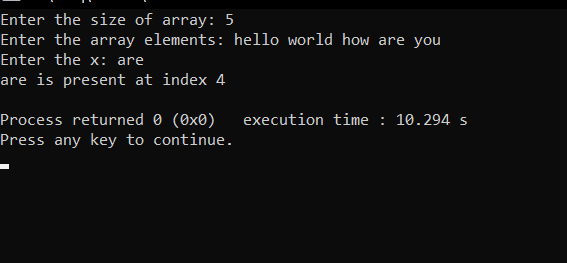
**Examples:**

**Input: arr[] = {”Hi”, ”Folks”, ”ide”, ”for”,”practice”}, x = ”ide”**

**Output: 2, The String x is present at index 2.**

**Input : arr[ ] = {”Hi”, ”Folks”, ”ide”, ”for”, ”practic”}, x = ”zz”**

**Output: 0 (Hint: substring not present.)**

****

int main()

**{**

int n;

cout<<"Enter the size of array: ";

cin>>n;

string \* arr=new string[n];

cout<<"Enter the array elements: ";

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

{

cin>>arr[i];

}

cout<<"Enter the x: ";

string x;

cin>>x;

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

{

if(x==arr[i])

{

cout<<x<<" is present at index "<<i+1<<endl;

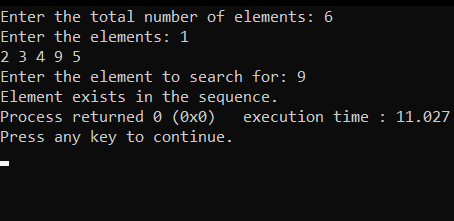
}

}

}

**Q8. Write a program using linear search to check whether the inputted**

**element belongs to it or not.**

 bool check(int val,vector<int> &arr)

{

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

{

if(val==arr[i]) return true;

}

return false;

}

int main()

{

int n;

cout<<"Enter the total number of elements: ";

cin>>n;

vector<int> arr;

int ele;

cout<<"Enter the elements: ";

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

{

cin>>ele;

arr.push\_back(ele);

}

int val;

cout<<"Enter the element to search for: ";

cin>>val;

if(check(val,arr))

{

cout<<"Element exists in the sequence.";

}

else{

cout<<"Element does not exist in the sequence.";

}

}