Heaven's Light Is Our Guide

Rajshahi University of Engineering & Technology

Department of Computer Science & Engineering

Course Code: CSE 2102

Course Title: Discrete Mathematics Sessional

Experiment No. 01

Submitted By	**Submitted To**
Name : Khandoker Sefayet Alam	Md. Azmain Yakin Srizon
Roll: 2003121	Lecturer
Section: C, Session: 2020-2021	Department of CSE, RUET

Submission Date: 25/05/23

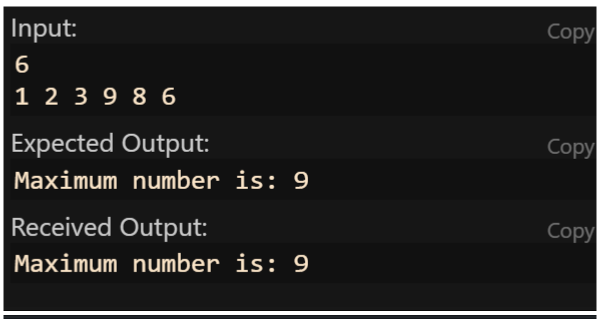
Problem Title : Finding the maximum number.

Description : We find the the maximum number by checking every element of the array.

Code:

```cpp
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

//VVI
#define fast ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
#define ll long long
int main()
{
    fast;
    ll n;
    cin>>n;
    vector<ll>vec(n);
    ll maxm=INT_MIN;
    for(ll i=0;i<n;i++){
        cin>>vec[i];
        maxm=max(maxm,vec[i]);
    }
    cout<<"Maximum number is: "<<maxm<<endl;
    return 0;
}
```

Output:



```
Input:
6
1 2 3 9 8 6
Expected Output:
Maximum number is: 9
Received Output:
Maximum number is: 9
```

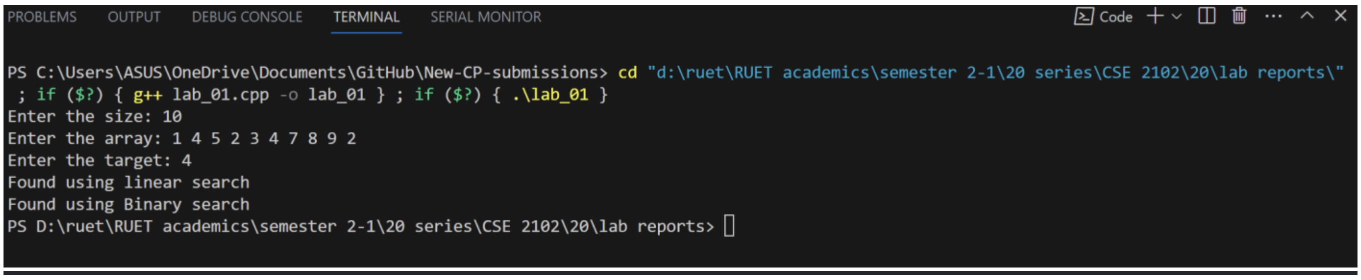
Problem Title : Linear Search, Binary Search

Problem Description : In linear Search you may end up checking all the elements to find out a particular element. But , in binary search we eliminate half of the elements of the array in every iteration so the searching become optimized & we end up finding our element faster.

Code:

```cpp
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

//VVI
#define fast ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
#define ll long long

bool linearSearch(vector<ll>&vec,ll n,ll tar){
    for(ll i=0;i<n;i++){
        if(vec[i]==tar){
            return true;
        }
    }
    return false;
}

bool BinarySearch(vector<ll>&vec,ll n,ll tar){
    ll l=0,r=n-1;
    ll mid;
    while(l<=r){
        mid=(l+r)/2;
        // cout<<l<<" "<<r<<" "<<mid<<endl;
        if(tar<vec[mid]) r=mid-1;
        else if(vec[mid]==tar){
            return true;
        }
        else l=mid+1;
    }
    return false;
}

int main()
{
    // fast;
    ll n;//size of the array
    cout<<"Enter the size: ";
    cin>>n;
    vector<ll>vec(n);
    cout<<"Enter the array: ";
    for(ll i=0;i<n;i++){
        cin>>vec[i];
    }
    ll target;
    cout<<"Enter the target: ";
    cin>>target;
    sort(vec.begin(),vec.end());

    if(linearSearch(vec,n,target)){
        cout<<"Found using linear search"<<endl;
    }
    else cout<<"Not found using linear search"<<endl;
    if(BinarySearch(vec,n,target)){
        cout<<"Found using Binary search"<<endl;
    }
    else cout<<"Not found using  Binary search"<<endl;
    return 0;
}
```

Output:



```
PS C:\Users\ASUS\OneDrive\Documents\GitHub\New-CP-submissions> cd "d:\ruet\RUET academics\semester 2-1\20 series\CSE 2102\20\lab reports\"
 ; if ($?) { g++ lab_01.cpp -o lab_01 } ; if ($?) { .\lab_01 }
Enter the size: 10
Enter the array: 1 4 5 2 3 4 7 8 9 2
Enter the target: 4
Found using linear search
Found using Binary search
PS D:\ruet\RUET academics\semester 2-1\20 series\CSE 2102\20\lab reports>
```

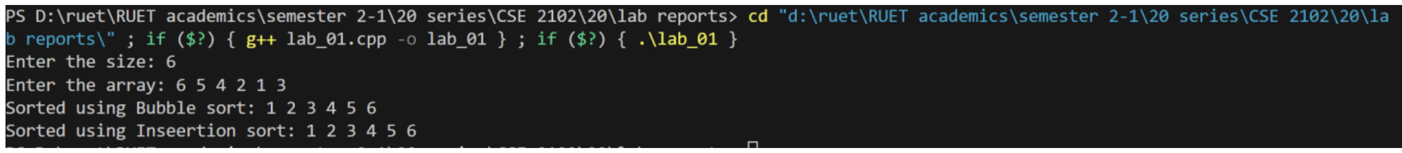
Problem: Bubble Sort and Insertion Sort.

Description: Two nested for loop is used to implement bubble sort and insertion sort algorithm as we need two compere all the elements for each of the elements in those algorithm.

Code:

```cpp
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

//VVI
#define fast ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
#define ll long long

void insertionSort(vector<ll>&vec,ll n)
{
    ll i,key,j;
    for (i = 1;i<n;i++)
    {
        key=vec[i];
        j = i - 1;
        while (j >= 0 && vec[j]>key)
        {
            vec[j+1]=vec[j];
            j=j-1;
        }
        vec[j+1]=key;
    }
}


int main()
{
    // fast;
    ll n;//size of the array
    cout<<"Enter the size: ";
    cin>>n;
    vector<ll>vec(n);
    vector<ll>vec2(n);
    cout<<"Enter the array: ";
    for(ll i=0;i<n;i++){
        cin>>vec[i];
        vec2[i]=vec[i];
    }
    for(ll i=0;i<n;i++){
        for(ll j=0;j<i;j++){
            if(vec[j]>vec[i]){
                swap(vec[j],vec[i]);
            }
        }
    }
    cout<<"Sorted using Bubble sort: ";
    for(ll i=0;i<n;i++){
        cout<<vec[i]<<" ";
    }
    cout<<endl;
    cout<<"Sorted using Inseertion sort: ";
    insertionSort(vec2,n);
    for(ll i=0;i<n;i++){
        cout<<vec2[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```

Output:



```
PS D:\ruet\RUET academics\semester 2-1\20 series\CSE 2102\20\lab reports> cd "d:\ruet\RUET academics\semester 2-1\20 series\CSE 2102\20\la
b reports\" ; if ($?) { g++ lab_01.cpp -o lab_01 } ; if ($?) { .\lab_01 }
Enter the size: 6
Enter the array: 6 5 4 2 1 3
Sorted using Bubble sort: 1 2 3 4 5 6
Sorted using Inseertion sort: 1 2 3 4 5 6
```

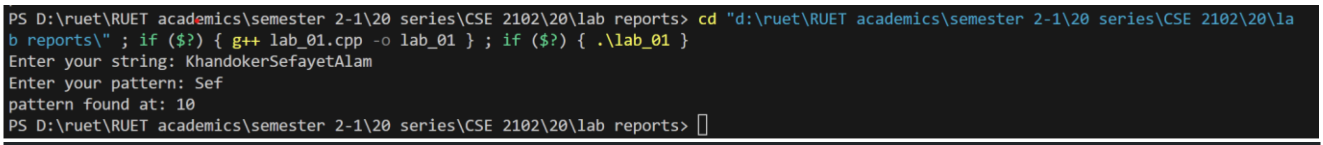
Problem: Naive String Matcher

Discription: Naive string matching is a simple algorithm used to find occurrences of a pattern within a text. It works by comparing the pattern against all possible shifts of the text, character by character.

Code:

```
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

//VVI
#define fast ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
#define ll long long

ll NaiveStringMatch(string s,string pat){
    ll m=s.size();
    ll n=pat.size();
    for(ll i=0;i<m;i++){
        ll j=0;
        while(j<n&& s[i+j]==pat[j]){
            j++;
            if(j==n){
                return i+1;
            }
        }
    }
    return -1;
}
int main()
{
    // fast;
    string s;
    cout<<"Enter your string: ";
    cin>>s;
    string pat;
    cout<<"Enter your pattern: ";
    cin>>pat;
    ll pos=NaiveStringMatch(s,pat);
    if(pos==-1) cout<<"Not Found"<<endl;
    else{
        cout<<"pattern found at: "<<pos<<endl;
    }
    return 0;
}
```

Output:



```
PS D:\ruet\RUET academics\semester 2-1\20 series\CSE 2102\20\lab reports> cd "d:\ruet\RUET academics\semester 2-1\20 series\CSE 2102\20\la
b reports\" ; if ($?) { g++ lab_01.cpp -o lab_01 } ; if ($?) { .\lab_01 }
Enter your string: KhandokerSefayetAlam
Enter your pattern: Sef
pattern found at: 10
PS D:\ruet\RUET academics\semester 2-1\20 series\CSE 2102\20\lab reports> 
```

Problem: Cashier Problem

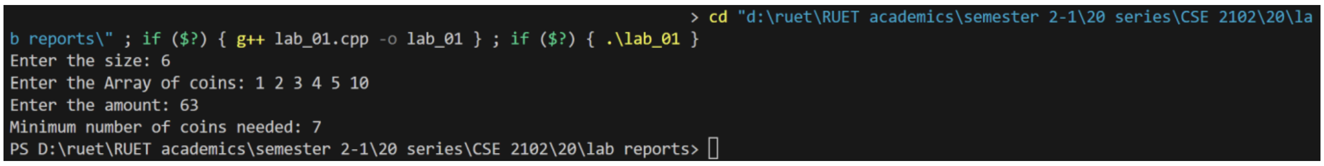
Discription: The problem can be stated as follows: Given a set of available denominations (e.g., coins or bills) and a target amount, the goal is to find the minimum number of denominations needed to make change for the target amount.

Code:

```
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;

//VVI
#define fast ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
#define ll long long
```

```cpp
int main()
{
    // fast;
    ll n;
    cout<<"Enter the size: ";
    cin>>n;
    vector<ll>vec(n);
    cout<<"Enter the Array of coins: ";
    for(ll i=0;i<n;i++){
        cin>>vec[i];
    }
    ll amount;
    cout<<"Enter the amount: ";
    cin>>amount;
    //sort the array in descending order
    ll ans=0;
    sort(vec.rbegin(),vec.rend());
    for(ll i=0;i<n;i++){
        while(vec[i]<=amount){
            amount-=vec[i];
            ans++;
        }
    }
    cout<<"Minimum number of coins needed: "<<ans<<endl;
    return 0;
}
```

Output:



```
                                                                  > cd "d:\ruet\RUET academics\semester 2-1\20 series\CSE 2102\20\la
b reports\" ; if ($?) { g++ lab_01.cpp -o lab_01 } ; if ($?) { .\lab_01 }
Enter the size: 6
Enter the Array of coins: 1 2 3 4 5 10
Enter the amount: 63
Minimum number of coins needed: 7
PS D:\ruet\RUET academics\semester 2-1\20 series\CSE 2102\20\lab reports> 
```