Walchand College of Engineering, Sangli
Computer Science & Engineering
Third Year

Course: Design and analysis of algorithm Lab
Lab course coordinator:
Dr. B. F. Momin- Batch: - T6, T7, T8
Mr. Kiran P. Kamble- Batch: - T1, T2, T3, T4, T5

Week 1 Assignment

Part: 1

Sorting Algorithm

Name: Trupti Rajendra Patil

Prn: 2020BTECS00051

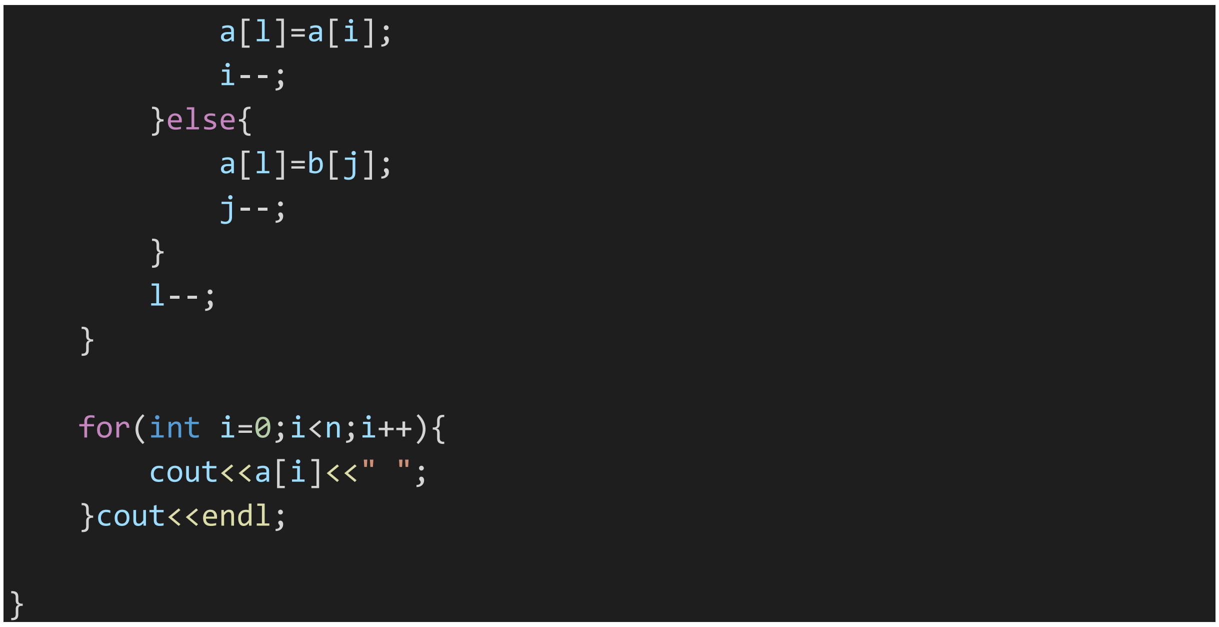
Q) You are given two sorted array, A and B, where A has a large enough buffer at the end to hold B. Write a method to merge B into A in sorted order.

Algorithm:

1. Initialize variable i at end of array a where numbers are present, j at end of array b and l at end of array a
2. Now traverse both the array.
3. If element of array a is greater than element of array b then insert it at last position and decrement l.
4. Otherwise, if element of array b is greater than or equal to element of array a than insert it at last position and decrement l.

Code:

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n=9,m=4;
    int a[n]={9,11,15,24,50},b[m]={2,7,11,23};

    int i=n-m-1,j=m-1,l=n-1;

    while(j>=0){
        if(i>=0 && a[i]>b[j]){
```



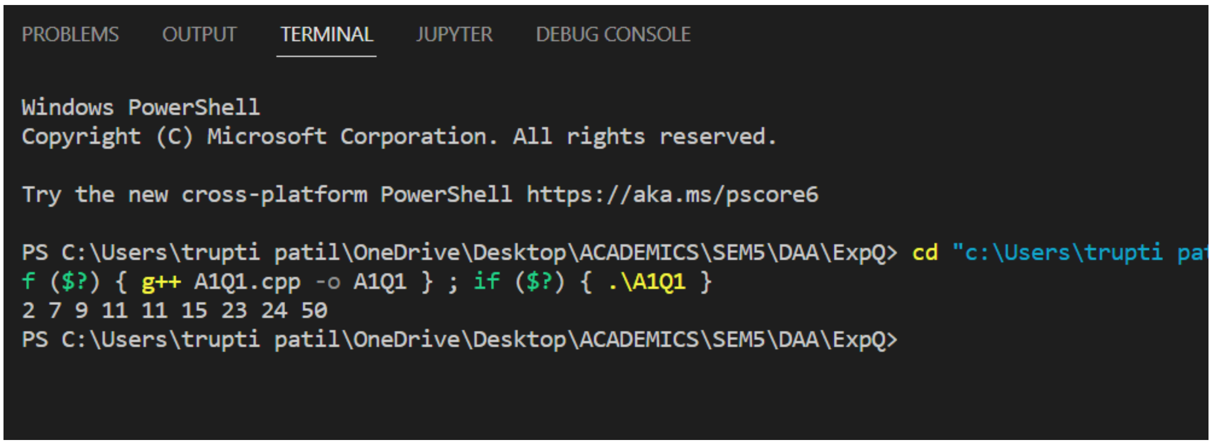
```
            a[l]=a[i];
            i--;
        }else{
            a[l]=b[j];
            j--;
        }
        l--;
    }

    for(int i=0;i<n;i++){
        cout<<a[i]<<" ";
    }cout<<endl;

}
```

Complexity Analysis:

Time Complexity: O(m+n)

Space complexity: O(1)

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> cd "c:\Users\trupti pa
f ($?) { g++ A1Q1.cpp -o A1Q1 } ; if ($?) { .\A1Q1 }
2 7 9 11 11 15 23 24 50
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ>
```

Q) Write a method to sort an array of string so that all the anagrams are next to each other.

Algorithm:

Steps:

1. Store the strings and its index in vector of pair.
2. Sort each string in vector of pair.
3. Now sort the vector of pair.
4. Print the string according to index in vector of pair.

Code:

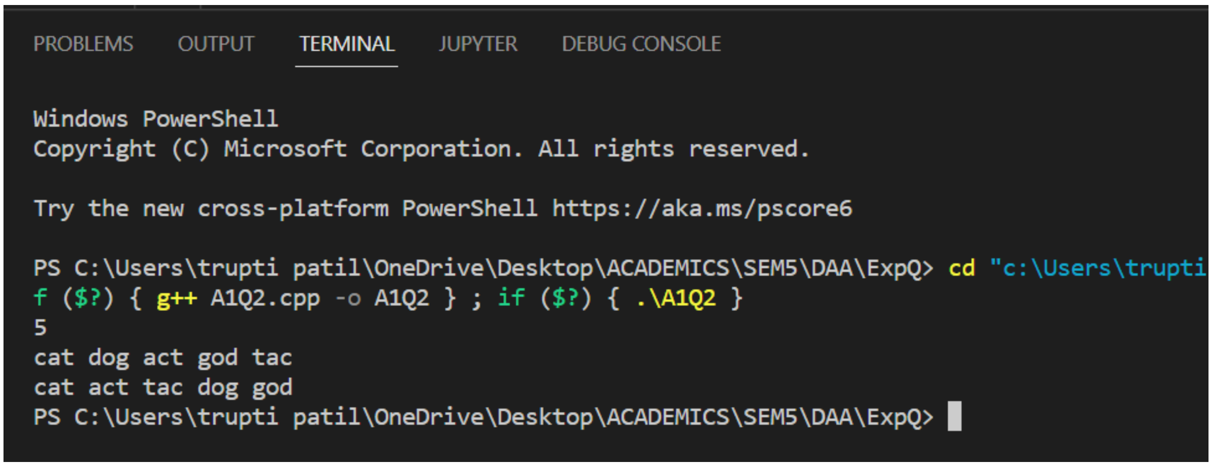
```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n;
    cin>>n;

    string arr[n];
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }

    vector<pair<string,int>> v;
    for(int i=0;i<n;i++){
        v.push_back({arr[i],i});
    }

    for(int i=0;i<v.size();i++){
        sort(v[i].first.begin(),v[i].first.end());
    }

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

    string ans[n];
    for(int i=0;i<n;i++){
        ans[i]=arr[v[i].second];
    }

    for(int i=0;i<n;i++){
        cout<<ans[i]<<" ";
    }cout<<endl;
}
```

Complexity Analysis:

Time Complexity: O(nmLogm + mnLogn)

Space complexity: O(n)

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> cd "c:\Users\trupti
f ($?) { g++ A1Q2.cpp -o A1Q2 } ; if ($?) { .\A1Q2 }
5
cat dog act god tac
cat act tac dog god
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> █
```

Q) Given a sorted array of *n* integers that has been rotated an unknown number of times, write code to find an element in the array. You may assume that the array was originally sorted in increasing order.

EXAMPLE

Input: find 5 in {15, 16, 19, 20, 25, 1, 3, 4, 5, 7, 10, 14}

Output: 8 (the index of 5 in the array)

Algorithm:

The idea is to find the pivot point, divide the array into two sub-arrays and perform a binary search.

The main idea for finding a pivot is –

For a sorted (in increasing order) and rotated array, the pivot element is the only element for which the next element to it is smaller than it.

Using binary search based on the above idea, pivot can be found.

It can be observed that for a search space of indices in the range [l, r] where the middle index is mid,

If rotation has happened in the left half, the element at l will obviously be greater than the one at mid.

Otherwise, the left half will be sorted but the element at mid will be greater than the one at r. After the pivot is found divide the array into two sub-arrays.

Now the individual sub-arrays are sorted so the element can be searched using Binary Search.

Binary Search

The basic steps to perform Binary Search are:

1. Begin with the mid element of the whole array as a search key.

2. If the value of the search key is equal to the item then return an index of the search key.

3. Or if the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half.

4. Otherwise, narrow it to the upper half.

5. Repeatedly check from the second point until the value is found or the interval is empty.

Code:

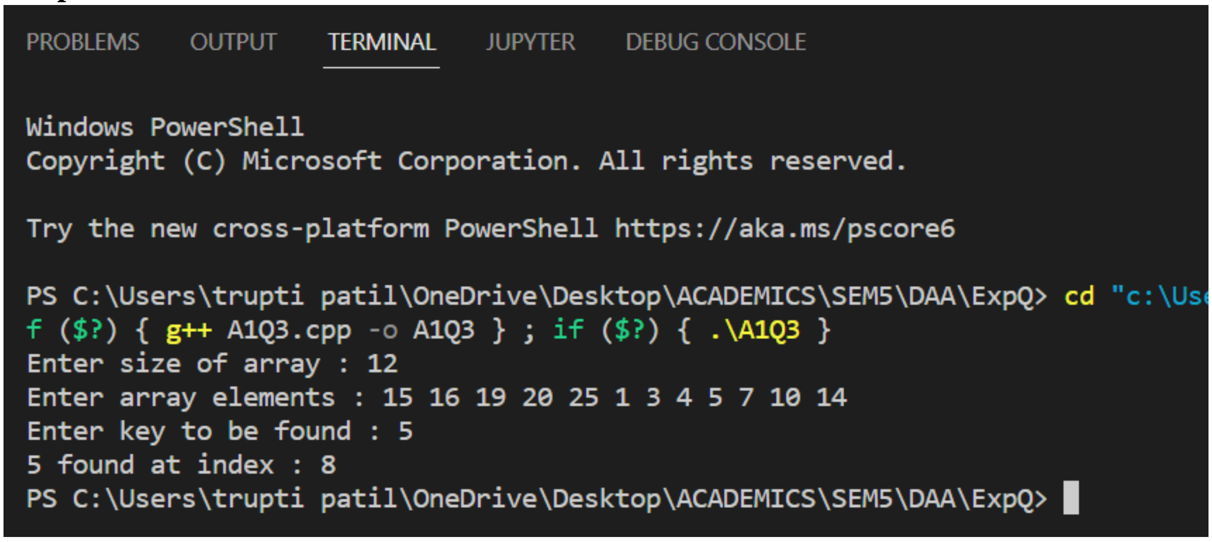
```cpp
#include <iostream>
#include <vector>
using namespace std;
int binarySearch(int key, int a[], int l, int h)
{
    int mid = (l + h) / 2;
    if (h < l)
        return -1;
    if (a[mid] == key)
        return mid;
    else if (a[mid] < key)
        return binarySearch(key, a, mid + 1, h);
    else
        return binarySearch(key, a, l, mid - 1);
}

int search(int a[], int n, int target)
{
    int x = 0, ans = 0;
    bool temp = false;
    for (int i = 0; i < n - 1; i++)
    {
        if (a[i] > a[i + 1])
        {
            x = i + 1;
            temp = true;
            break;
        }
    }
    if (temp == true)
    {
        if (target <= a[n - 1])
            ans = binarySearch(target, a, x, n - 1);
        else
            ans = binarySearch(target, a, 0, x - 1);
    }
    else
        ans = binarySearch(target, a, 0, n - 1);

    return ans;
}
int main()
{
    int n, element;
    cout << "Enter size of array: ";
    cin >> n;
    int a[n];
    cout << "Enter elements of array: ";
    for(int i = 0; i < n; i++)
        cin >> a[i];
    cout << "Enter a element to search: ";
    cin >> element;
    int ans;
    ans = search(a, n, element);
    if(ans == -1)
        cout << "Element is not present in array.";
    else
        cout << "Element is present at index " << ans <<
".\n";
    return 0;
}
```

Complexity Analysis:

Time Complexity: O(logn)

Space complexity: O(1)

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> cd "c:\Us
f ($?) { g++ A1Q3.cpp -o A1Q3 } ; if ($?) { .\A1Q3 }
Enter size of array : 12
Enter array elements : 15 16 19 20 25 1 3 4 5 7 10 14
Enter key to be found : 5
5 found at index : 8
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ>
```

Q) Imagine you have a 20GB file with one string per line. Explain how you would sort the file.

This is accomplish by external sort. Bring only part of the data into memory at x megabytes each. Sort each chunks separately and saved back to the file system. Once all chunks are sorted, merge the chunks, one by one.

Q) Given a sorted array of string which is interspersed with empty string, write a method to find the location of a given string.

EXAMPLE

Input: find “ball” in {“at”, “”, “”, “ball”, “”, “”, “car”, “”, “”, “dad”, “”,””}

Output: 4

Algorithm:

1. Like normal binary search, we compare given str with middle string.

2. If middle string is empty, we find the closest non-empty string x (by linearly searching on both sides).

3. Once we find x, we do standard binary search, i.e., we compare given str with x.

4. If str is same as x, we return index of x.

5. If str is greater, we recur for right half, else we recur for left half.

Code:

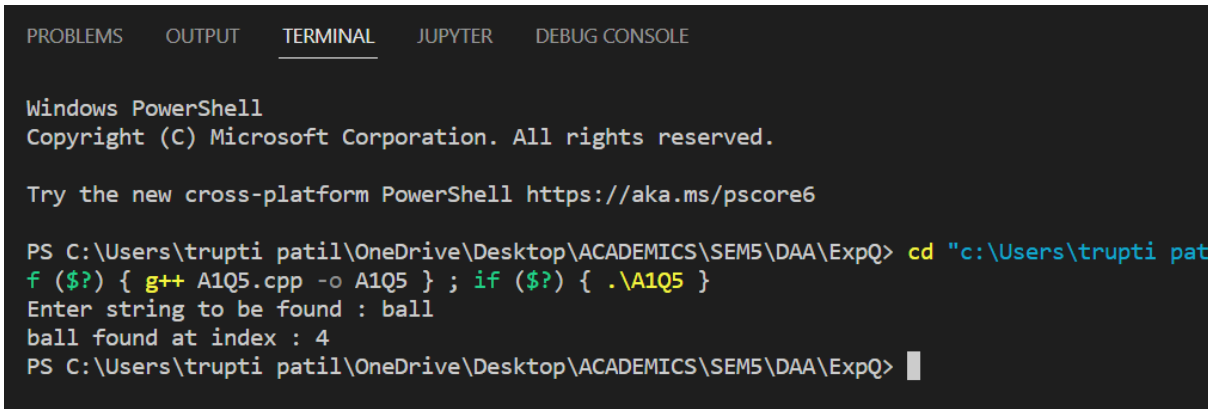
```
#include<bits/stdc++.h>
using namespace std;

int Search(string arr[],int n,int l,int r,string str){
    if(l>r){
        return -1;
    }

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

    if(arr[mid]==""){
        int l1=mid-1,r1=mid+1;

        while(l1>=l && r1<=r){
            if(l1<l && r1>r){
                return -1;
            }

            if(r1<=r && arr[r1]!=""){
                mid=r1;
                break;
            }

            if(l1>=l && arr[l1]!=""){
                mid=l1;
                break;
            }
            l1--;
            r1++;
        }
    }

    if(str.compare(arr[mid])==0){
        return mid;
    }else if(str.compare(arr[mid])<0){
        return Search(arr,n,l,mid-1,str);
    }
    return Search(arr,n,mid+1,r,str);
}

int main(){
    int n=12;
    string arr[n]={"at","","","ball","","","car","","","dad","",""};

    string str;
    cout<<"Enter string to be found : ";
    cin>>str;

    int l=0,r=n-1;
    int index=Search(arr,n,l,r,str);
    cout<<str<<" found at index : "<<index+1<<endl;
}
```

Complexity Analysis:

Time Complexity: O(logn)

Space complexity: O(1)

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> cd "c:\Users\trupti pat
f ($?) { g++ A1Q5.cpp -o A1Q5 } ; if ($?) { .\A1Q5 }
Enter string to be found : ball
ball found at index : 4
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> 
```

Q) Given an M*N matrix in which each row and each column is sorted in ascending order, write a method to find an element.

Algorithm:

1. Create two variable i = 0, j = m-1 as index of row and column and key is element to be found.
2. Run a loop until i = n and j>=0
3. Check if the current element is greater than Key then decrease the count of j. Exclude the current column.

4. Check if the current element is less than x then increase the count of i. Exclude the current row.
5. If the element is equal, then print the position and end.

Code:

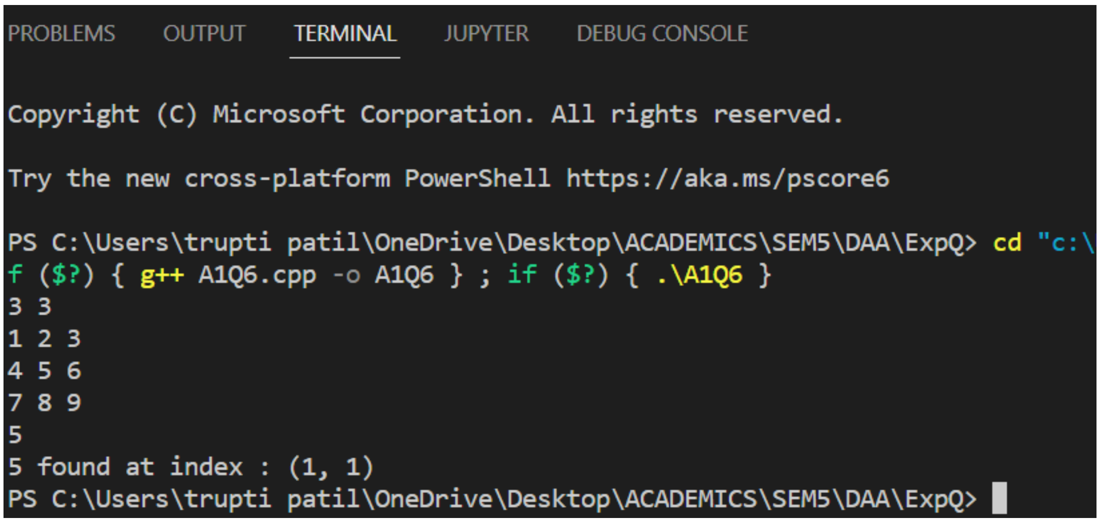
```
#include<bits/stdc++.h>
using namespace std;


int main(){
    int n,m;
    cin>>n>>m;

    int arr[n][m];
    for(int i=0;i<n;i++){
        for(int j=0;j<m;j++){
            cin>>arr[i][j];
        }
    }

    int key;
    cin>>key;

    int i=0,j=m-1;
    if(key<arr[0][0] || key>arr[n-1][m-1]){
        return -1;
    }

    bool ans=false;
    while(i<n && j>=0){
        if(arr[i][j]==key){
            ans=true;
            cout<<key<<" found at index : ("<<i<<", "<<j<<")"<<endl;
            break;
        }else if(arr[i][j]>key){
            j--;
        }else{
            i++;
        }
    }

    if(ans==false){
        cout<<key<<" not found "<<endl;
    }
}
```

Complexity Analysis:

Time Complexity: O(n+m)

Space complexity: O(1)

Output:



```
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> cd "c:\
f ($?) { g++ A1Q6.cpp -o A1Q6 } ; if ($?) { .\A1Q6 }
3 3
1 2 3
4 5 6
7 8 9
5
5 found at index : (1, 1)
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> 
```

Q) A circus is designing a tower routine consisting of people standing atop one another's shoulders. For practical and aesthetic reasons, each person must be both shorter and lighter than the person below him or her. Given the heights and weight of each circus, write a method to compute the largest possible number of people in such tower.

EXAMPLE:

Input(ht,wt): (65, 100) (70, 150) (56, 90) (75,190) (60, 95) (68, 110).

Output: The longest tower is length 6 and includes from top to bottom:

(56, 90) (60, 95) (65, 100) (68, 110) (70, 150) (75, 190)

Algorithm:

1. Store the height and weight of people in vector of pair.
2. Sort the vector of pair according to height of people and if their heights are equal then sort it by weight.
3. Now traverse the vector of pair till the weight of first person is less than next person and keep the track of start and end point.
4. If the weight of first person is greater than next, then initialize the start and end again.
5. Get the max length.

Code:

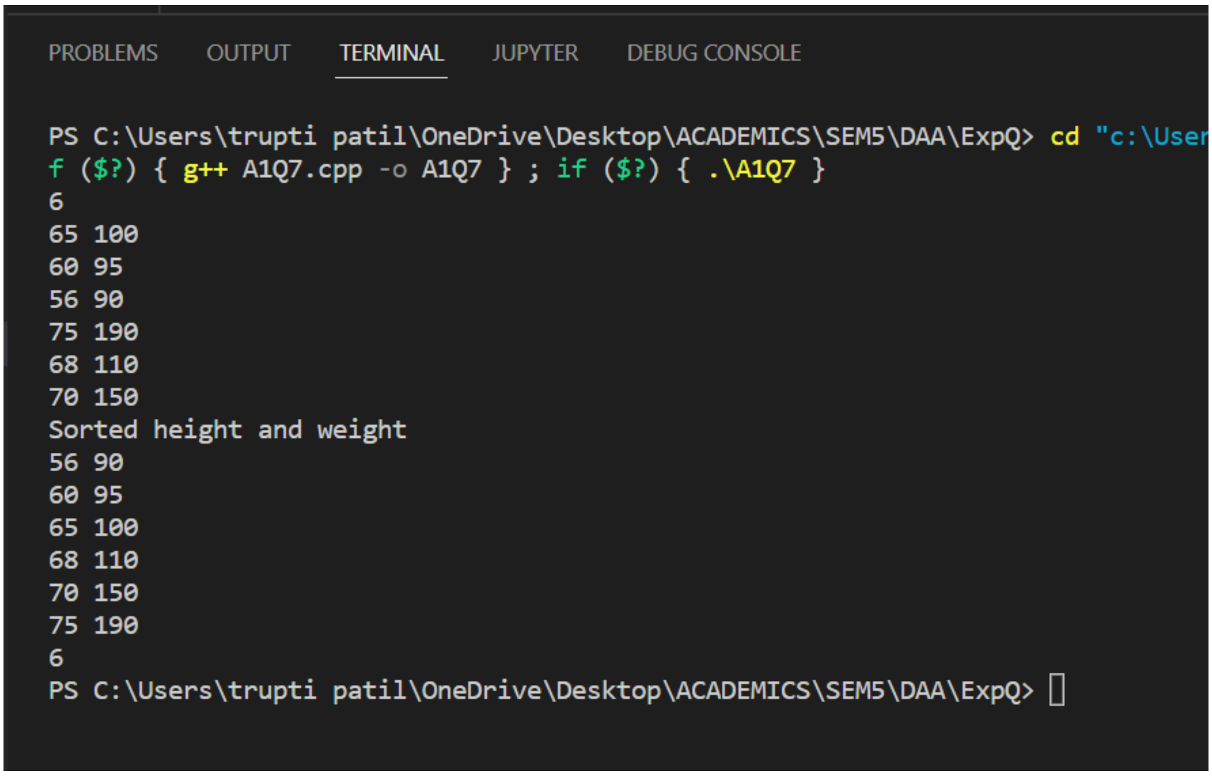
```cpp
#include<bits/stdc++.h>
using namespace std;

bool sorted(pair<int,int>& a,pair<int,int>& b){
    if(a.first==b.first){
        return a.second<b.second;
    }
    return a.first<b.first;
}

int main(){
    int n;
    cin>>n;

    int x,y;
    vector<pair<int,int>>v;

    for(int i=0;i<n;i++){
        cin>>x;
        cin>>y;
        v.push_back(make_pair(x,y));
    }

    sort(v.begin(),v.end(),sorted);
    cout<<"Sorted height and weight"<<endl;
    for(int i=0;i<v.size();i++){
        cout<<v[i].first<<" "<<v[i].second<<endl;
    }

    int start=0,end=0,maxPeople=0,ans=INT_MIN;
    for(int i=1;i<v.size();i++){
        if(v[i].second>v[i-1].second){
            end=i;
        }else{
            maxPeople=max(maxPeople,end-start+1);
            start=i;
            end=i;
        }
    }
     ans=maxPeople;
    cout<<ans<<endl;
}
```

Complexity Analysis:

Time Complexity: O(nlogn)

Space complexity: O(1)

Output:



```
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> cd "c:\User
f ($?) { g++ A1Q7.cpp -o A1Q7 } ; if ($?) { .\A1Q7 }
6
65 100
60 95
56 90
75 190
68 110
70 150
Sorted height and weight
56 90
60 95
65 100
68 110
70 150
75 190
6
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> 
```

Q) Imagine you are reading in stream of integers. Periodically, you wish to be able to look up the rank of number *x* (the number of values less than or equal to *x*). Implement the data structures and algorithms to support these operations. That is, Implement the method *track (int x)*, which is called when each number is generated, and the method *getRankOfNumber (int x)*, which return the number of values less than or equal to *x* (not including x itself).

EXAMPLE

Stream (in order of appearance) : 5, 1, 4, 4, 5, 9, 7, 13, 3

getRankOfNumber(1) = 0

getRankOfNumber(3) = 1

getRankOfNumber(4) =3

Algorithm:

1. Traverse the array linearly to find rank of x
2. If arr[i]<=x increment count
3. If arr[i]==x decrese one from count before returning the answer
4. Since we are traversing linearly,one of the element is x itself if it is repeating

Code:

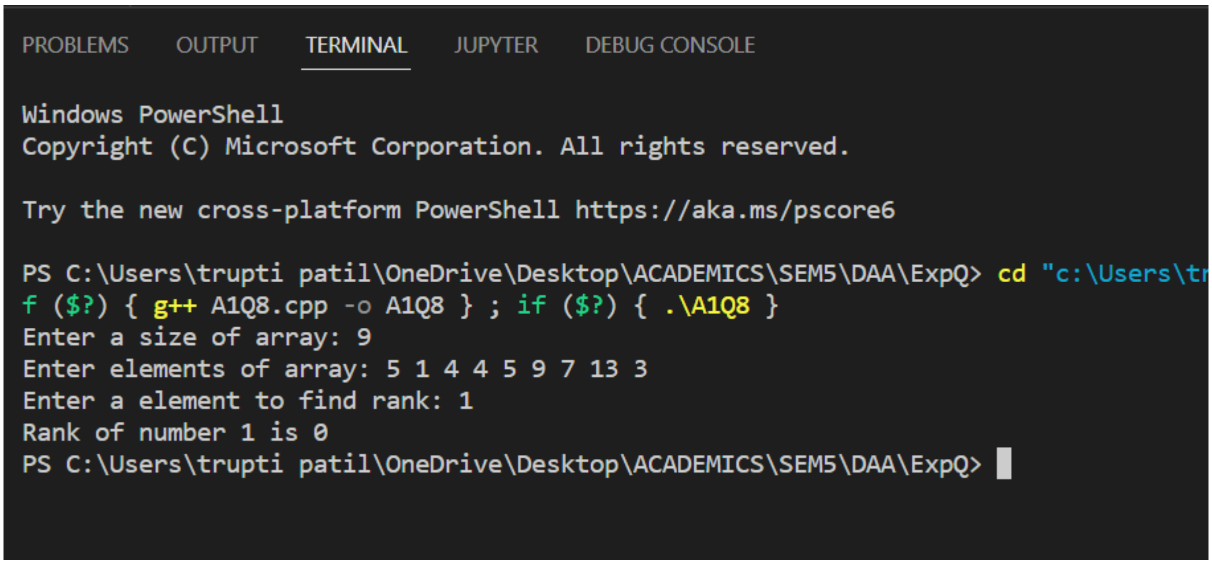
```
#include <bits/stdc++.h>
using namespace std;

int getRankOfNumber(int arr[], int n, int x)
{
    int count = 0;
    bool temp = false;
    for(int i = 0; i < n; i++)
    {
        if(arr[i] <= x)
            count++;
        if(arr[i] == x)
            temp = true;
    }
    if(temp == true)
        count--;
    return count;
}

int main()
{
    int n;
    cout << "Enter a size of array: ";
    cin >> n;
    int arr[n];
    cout << "Enter elements of array: ";
    for(int i = 0; i < n; i++)
        cin >> arr[i];
    int x;
    cout << "Enter a element to find rank: ";
    cin >> x;
    cout << "Rank of number " << x << " is " << getRankOfNumber(arr,
n, x);
    return 0;
}
```

Complexity Analysis:

Time Complexity: O(n)

Space complexity: O(1)

Output:



```
PROBLEMS   OUTPUT   TERMINAL   JUPYTER   DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> cd "c:\Users\tr
f ($?) { g++ A1Q8.cpp -o A1Q8 } ; if ($?) { .\A1Q8 }
Enter a size of array: 9
Enter elements of array: 5 1 4 4 5 9 7 13 3
Enter a element to find rank: 1
Rank of number 1 is 0
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ>
```