**Week 10**

**Q1) Given a list of activities with their starting time and finishing time. Your goal is to select maximum number of activities that can be performed by a single person such that selected activities must be non-conflicting. Any activity is said to be non-conflicting if starting time of an activity is greater than or equal to the finishing time of the other activity. Assume that a person can only work on a single activity at a time.**

**Input Format:**

**First line of input will take number of activities N.**

**Second line will take N space-separated values defining starting time for all the N activities.**

**Third line of input will take N space-separated values defining finishing time for all the N**

**activities.**

**Output Format:**

**Output will be the number of non-conflicting activities and the list of selected activities.**

**Solution:**

#include<bits/stdc++.h>

using namespace std;

int main() {

int i,n,s[n],f[n], e=INT\_MIN,c=0;

cin>>n;

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

cin>>s[i];

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

cin>>f[i];

vector<vector<int>> a;

vector<int> act;

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

a.push\_back({f[i],s[i],i+1});

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

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

{

if(a[i][1]>=e)

{

e=a[i][0];

c++;

act.push\_back(a[i][2]);

} }

cout<<"No. of non-conflicting activities : "<<c<<endl;

cout<<"List of selected activities : ";

for(i=0;i<act.size();i++)

cout<<act[i]<<",";

return 0;

}

**OUTPUT**

**Text

Description automatically generated**

**Q2) Given a long list of tasks. Each task takes specific time to accomplish it and each task has a deadline associated with it. You have to design an algorithm and implement it using a program to find maximum number of tasks that can be completed without crossing their deadlines and also find list of selected tasks.**

**Input Format:**

**First line will give total number of tasks n.**

**Second line of input will give n space-separated elements of array representing time taken by each task.**

**Third line of input will give n space-separated elements of array representing deadline associated with each task.**

**Output Format:**

**Output will be the total number of maximum tasks that can be completed.**

**Solution:**

#include<bits/stdc++.h>

using namespace std;

int main()

{

int n,i,t[n],f[n], e=INT\_MIN,c=0;

cin>>n;

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

cin>>t[i];

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

cin>>f[i];

vector<vector<int>> a;

vector<int> act;

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

a.push\_back({f[i],f[i]-t[i],i+1});

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

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

{

if(a[i][1]>=e)

{

e=a[i][0];

c++;

act.push\_back(a[i][2]);

} }

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

cout<<"Max number of tasks : "<<c<<endl;

cout<<"Selected task Numbers : ";

for(i=0;i<act.size();i++)

cout<<act[i]<<",";

return 0;

}

**OUTPUT**

**Text

Description automatically generated**

**Q3) Given an unsorted array of elements, design an algorithm and implement it using a program to find whether majority element exists or not. Also find median of the array. A majority element is an element that appears more than n/2 times, where n is the size of array.**

**Input Format:**

**First line of input will give size n of array.**

**Second line of input will take n space-separated elements of array.**

**Output Format:**

**First line of output will be 'yes' if majority element exists, otherwise print 'no'.**

**Second line of output will print median of the array.**

**Solution:**

#include<bits/stdc++.h>

using namespace std;

int main()

{

int n;

cin>>n;

int i,a[n],c,j;

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

cin>>a[i];

bool f=0;

sort(a,a+n);

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

{

c=1;

j=i+1;

while(j<n && a[j++]==a[i])

c++;

if(c>n/2)

{

cout<<"yes\n";

f=1;

break;

}

i=j-1;

}

if(f==0)

cout<<"no\n";

if(n%2!=0)

cout<<a[n/2];

else

cout<<((float)a[n/2]+a[n/2-1])/2;

return 0;

}

**OUTPUT**

**Text

Description automatically generated**