Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 1

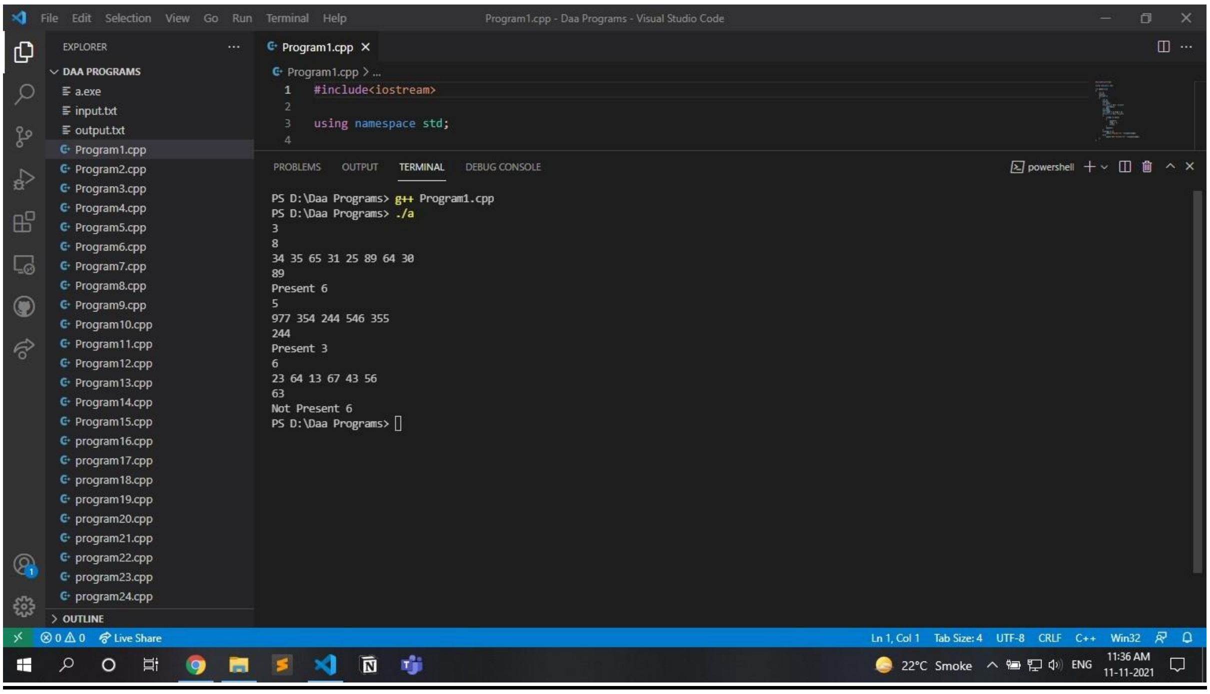
Given an array of nonnegative integers, design a linear algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for each input case.

Code:

```
#include<iostream>

using namespace std;

int main(void)
{
        int t;
        cin>>t;
        while(t--)
        {
                int n;
                cin>>n;
                int A[n];
                for(int i = 0;i < n;i++)
                        cin>>A[i];
                int key;
                cin>>key;
                int count = 0,flag = 0;
                for(int i = 0;i < n;i++)
                {
                        if(key == A[i])
                        {
                                count++;
                                flag = 1;
                                break;
                        }
                        count++;
                }
                if(flag == 1)
                        cout<<"Present"<<" "<<count<<endl;
                else
```

```
                        cout<<"Not Present"<<" "<<count<<endl;
          }
}
```

OUTPUT



```
#include<iostream>

using namespace std;

```

```
PS D:\Daa Programs> g++ Program1.cpp
PS D:\Daa Programs> ./a
3
8
34 35 65 31 25 89 64 30
89
Present 6
5
977 354 244 546 355
244
Present 3
6
23 64 13 67 43 56
63
Not Present 6
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 2

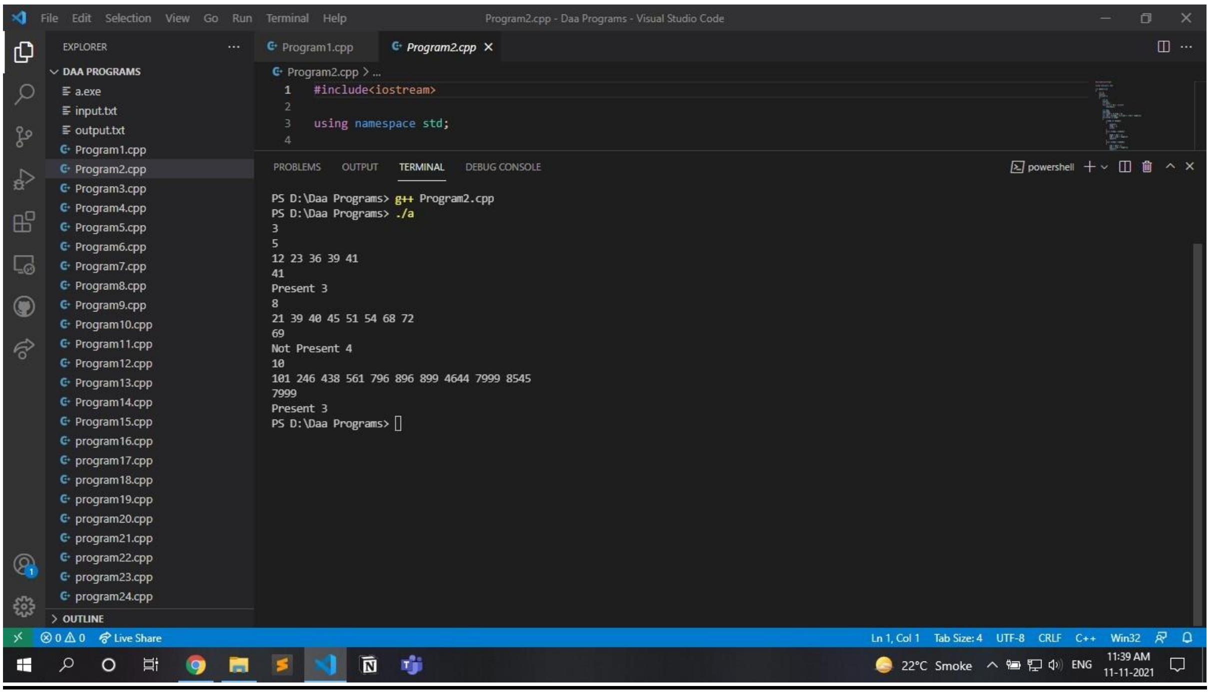
Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for each input case. (Time Complexity = O(nlogn), where n is the size of input).

Code:

```
#include<iostream>

using namespace std;

int main(void)
{
        int t;
        cin>>t;
        while(t--)
        {
                int n;
                cin>>n;
                int A[n];
                for(int i = 0;i < n;i++)
                        cin>>A[i];

                int key;
                cin>>key;
                int count = 0,flag = 0;
                int low = 0,high = n-1,mid = (low + high)/2;
                while(low <= high)
                {
                        if(key == A[mid])
                        {
                                count++;
                                flag = 1;
                                break;
                        }
                        else if(key < A[mid])
                        {
```

```
                high = mid - 1;
                mid = (low + high)/2;
                count++;
            }
            else if(key > A[mid])
            {
                low = mid + 1;
                mid = (low + high)/2;
                count++;
            }
        }
        if(flag == 1)
            cout<<"Present"<<" "<<count<<endl;
        else
            cout<<"Not Present"<<" "<<count<<endl;
    }
}
```

OUTPUT



```
#include<iostream>

using namespace std;

```

```
PS D:\Daa Programs> g++ Program2.cpp
PS D:\Daa Programs> ./a
3
5
12 23 36 39 41
41
Present 3
8
21 39 40 45 51 54 68 72
69
Not Present 4
10
101 246 438 561 796 896 899 4644 7999 8545
7999
Present 3
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

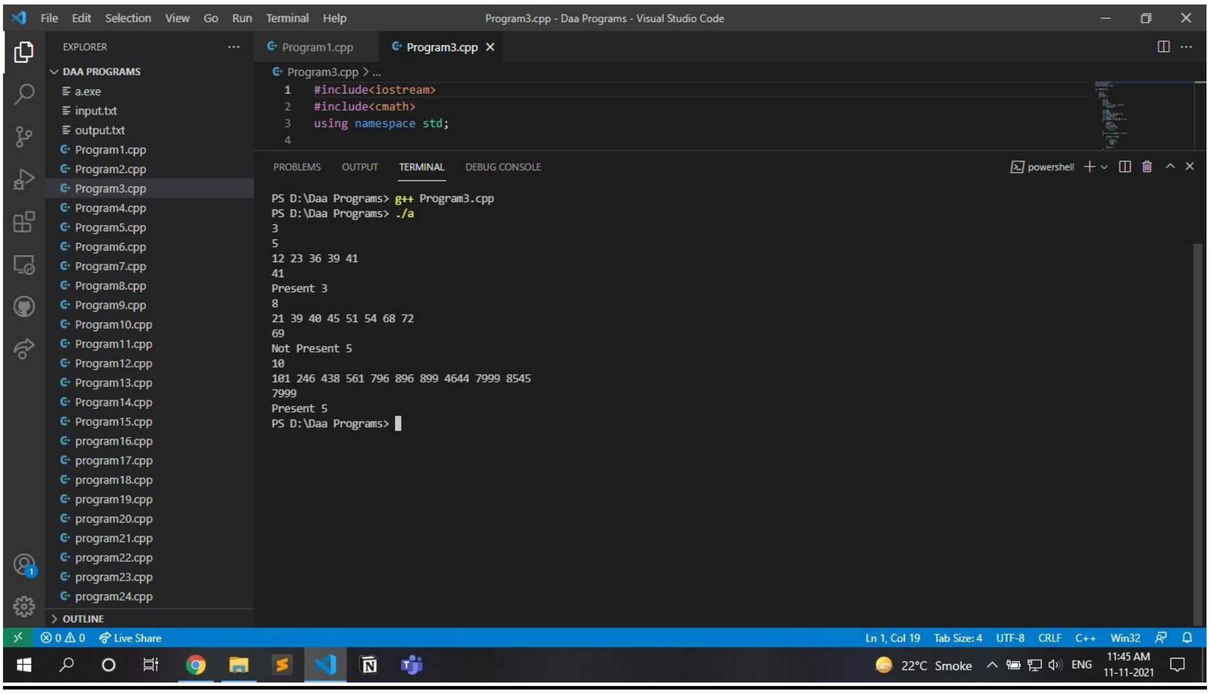
Program 3

Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether a given key element is present in the sorted array or not. For an array arr[n], search at the indexes arr[0], arr[2], arr[4],. ,arr[2k] and so on. Once the interval (arr[2k] < key < arr[2k+1]) is found, perform a linear search operation from the index 2k to find the element key. (Complexity < O(n), where n is the number of elements need to be scanned for searching)

Code:

```
#include<iostream>
#include<cmath>
using namespace std;

int main(void)
{
        int t;
        cin>>t;
        while(t--)
        {
                int n;
                cin>>n;
                int A[n];
                for(int i = 0;i < n;i++)
                        cin>>A[i];

                int key;
                cin>>key;
                int count = 0,flag = 0;
                int m = sqrt(n);
                int start = 0;
                while(A[m] <= key && m < n)
                {
                        count++;
                        start = m;
                        m += sqrt(n);
                        if(m > n - 1)
                                m = n;
                }
        for(int i = start;i < m;i++)
        {
            if(A[i] == key)
            {
                count++;
                flag = 1;
                break;
            }
            count++;
        }
        if(flag == 1)
            cout<<"Present"<<" "<<count<<endl;
        else
            cout<<"Not Present"<<" "<<count<<endl;
    }
}
```

OUTPUT



```
#include<iostream>
#include<cmath>
using namespace std;
```

```
PS D:\Daa Programs> g++ Program3.cpp
PS D:\Daa Programs> ./a
3
5
12 23 36 39 41
41
Present 3
8
21 39 40 45 51 54 68 72
69
Not Present 5
10
101 246 438 561 796 896 899 4644 7999 8545
7999
Present 5
PS D:\Daa Programs>
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 4

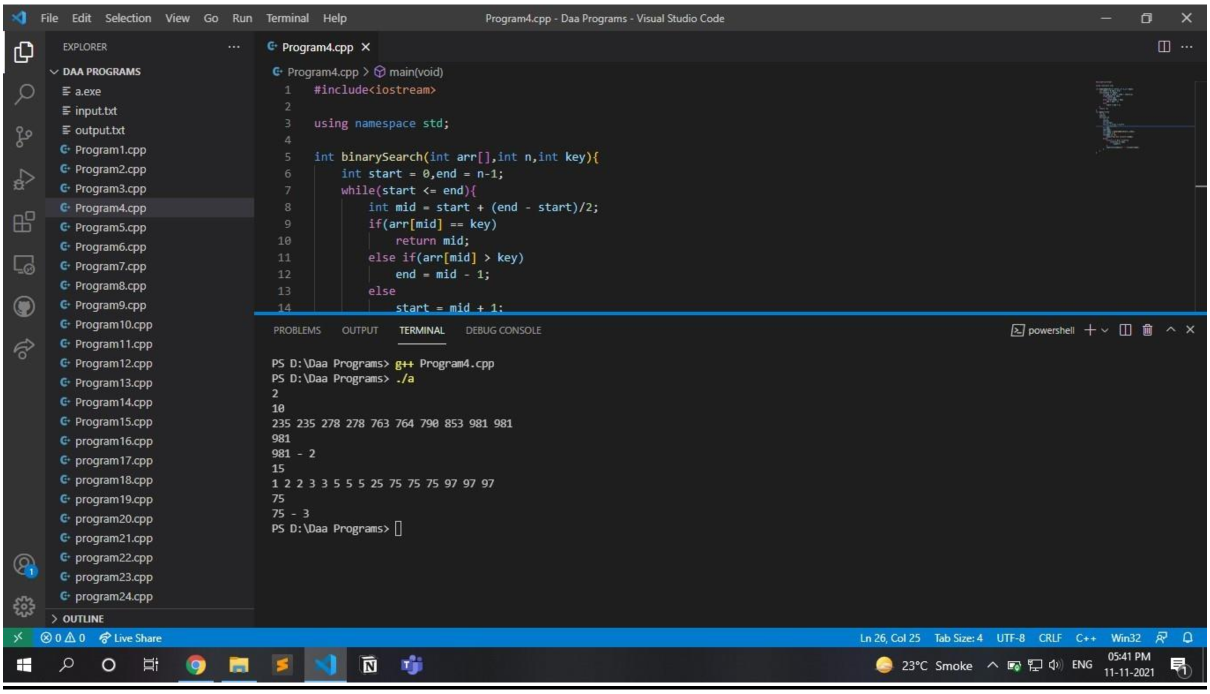
Given a sorted array of positive integers containing few duplicate elements, design an algorithm and implement it using a program to find whether the given key element is present in the array or not. If present, then also find the number of copies of given key. (Time Complexity = O(log n))

Code:

```
#include<iostream>

using namespace std;

int binarySearch(int arr[],int n,int key){
	int start = 0,end = n-1;
	while(start <= end){
		int mid = start + (end - start)/2;
		if(arr[mid] == key)
			return mid;
		else if(arr[mid] > key)
			end = mid - 1;
		else
			start = mid + 1;
	}
	return -1;
}
int main(void){
	int t;
	cin>>t;
	while(t--){
		int n;
		cin>>n;
		int arr[n];
		for(int i = 0;i < n;i++)
			cin>>arr[i];
		int key;
		cin>>key;
		int index = binarySearch(arr,n,key);
		int count = 0;
		if(index == -1)
```

```
                cout<<"Key Not present"<<endl;
            else{
                for(int i = 0;i < n;i++){
                    if(arr[i] == key)
                        count++;
                }
                cout<<arr[index]<<" - "<<count<<endl;
            }
        }
}
```

OUTPUT



```
#include<iostream>

using namespace std;

int binarySearch(int arr[],int n,int key){
    int start = 0,end = n-1;
    while(start <= end){
        int mid = start + (end - start)/2;
        if(arr[mid] == key)
            return mid;
        else if(arr[mid] > key)
            end = mid - 1;
        else
            start = mid + 1;
```

```
PS D:\Daa Programs> g++ Program4.cpp
PS D:\Daa Programs> ./a
2
10
235 235 278 278 763 764 790 853 981 981
981
981 - 2
15
1 2 2 3 3 5 5 5 25 75 75 75 97 97 97
75
75 - 3
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

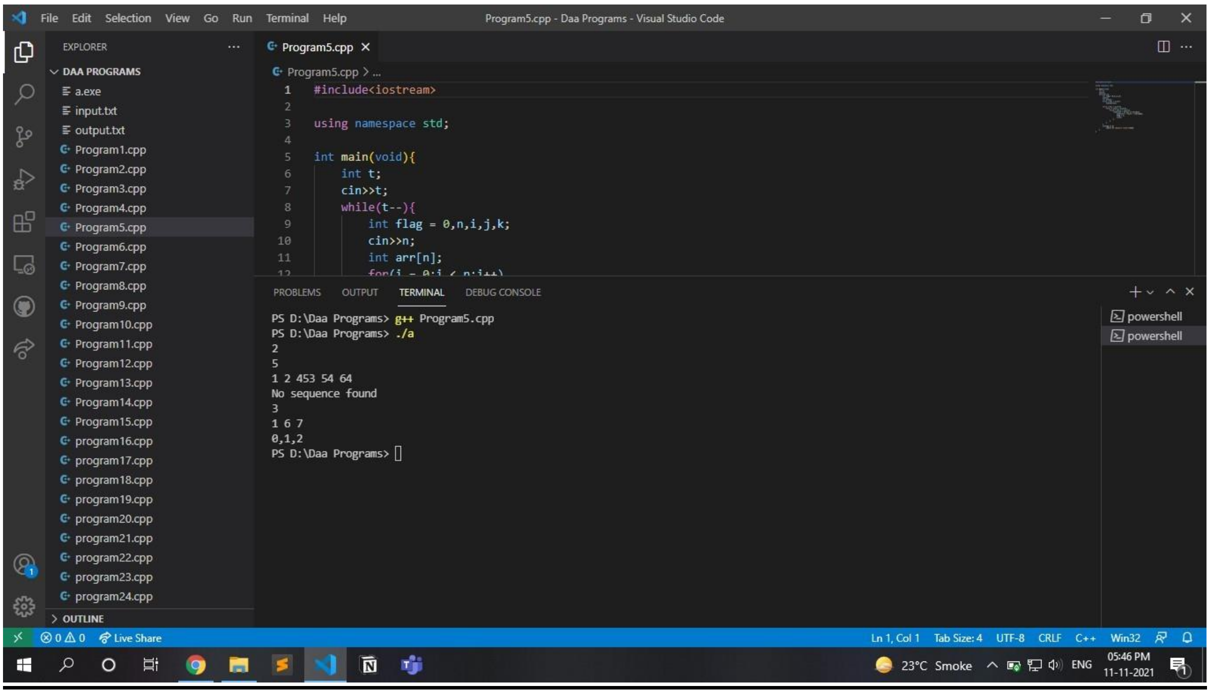
Program 5

Given a sorted array of positive integers, design an algorithm and implement it using a program to find three indices i, j, k such that arr[i] + arr[j] = arr[k].

Code:

```
#include<iostream>

using namespace std;

int main(void){
    int t;
    cin>>t;
    while(t--){
        int flag = 0,n,i,j,k;
        cin>>n;
        int arr[n];
        for(i = 0;i < n;i++)
            cin>>arr[i];

        for(i = 0;i < n;i++){
            for(j = i+1;j < n;j++){
                for(k = j+1;j < n;j++){
                    if(arr[i] + arr[j] == arr[k]){
                        cout<<i<<","<<j<<","<<k<<endl;
                        flag = 1;
                        break;
                    }
                }
            }
        }
        if(flag == 0)
            cout<<"No sequence found"<<endl;
    }
}
```

OUTPUT



```cpp
#include<iostream>

using namespace std;

int main(void){
    int t;
    cin>>t;
    while(t--){
        int flag = 0,n,i,j,k;
        cin>>n;
        int arr[n];
        for(i = 0;i < n;i++)
```

```
PS D:\Daa Programs> g++ Program5.cpp
PS D:\Daa Programs> ./a
2
5
1 2 453 54 64
No sequence found
3
1 6 7
0,1,2
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

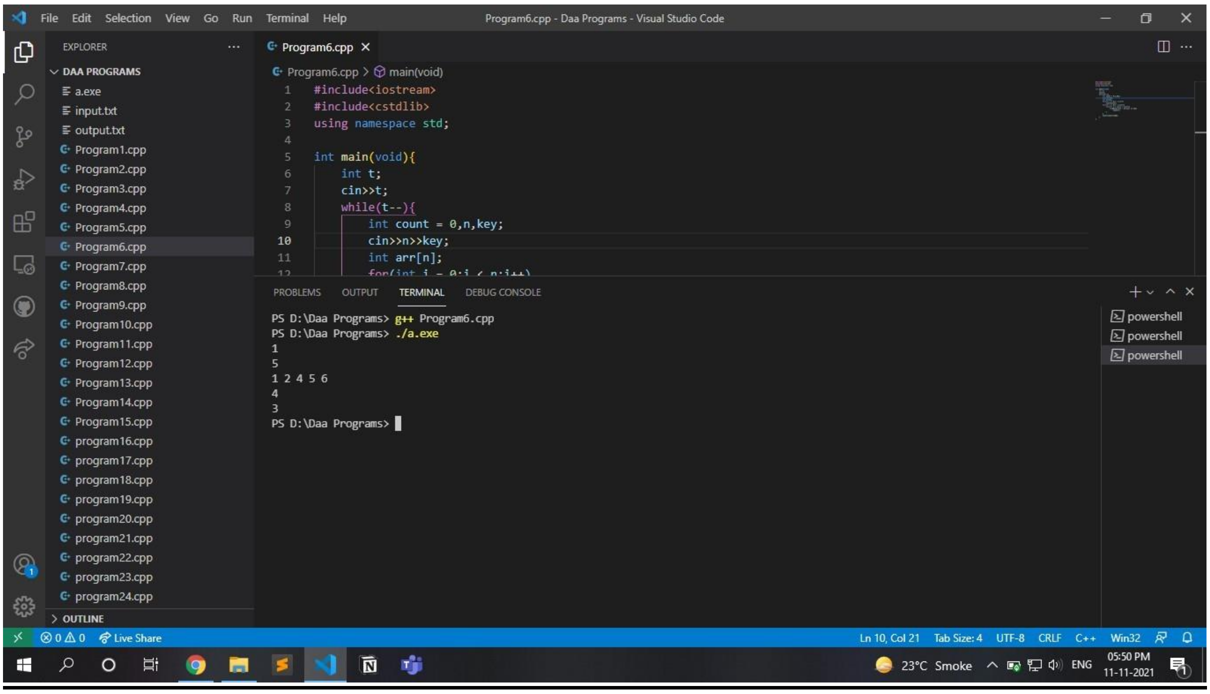
Program 6

Given an array of non-negative integers, design an algorithm and a program to count the number of pairs of integers such that their difference is equal to a given key, K.

Code:

```
#include<iostream>
#include<cstdlib>
using namespace std;

int main(void){
        int t;
        cin>>t;
        while(t--){
                int count = 0,n,key;
                cin>>n>>key;
                int arr[n];
                for(int i = 0;i < n;i++)
                        cin>>arr[i];
                for(int i = 0;i < n;i++){
                        for(int j = i+1;j < n;j++){
                                if(abs(arr[i] - arr[j]) == key)
                                        count++;
                        }
                }
                cout<<count<<endl;
        }
}
```

OUTPUT



```cpp
#include<iostream>
#include<cstdlib>
using namespace std;

int main(void){
    int t;
    cin>>t;
    while(t--){
        int count = 0,n,key;
        cin>>n>>key;
        int arr[n];
        for(int i = 0;i < n;i++)
```

```
PS D:\Daa Programs> g++ Program6.cpp
PS D:\Daa Programs> ./a.exe
1
5
1 2 4 5 6
4
3
PS D:\Daa Programs>
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

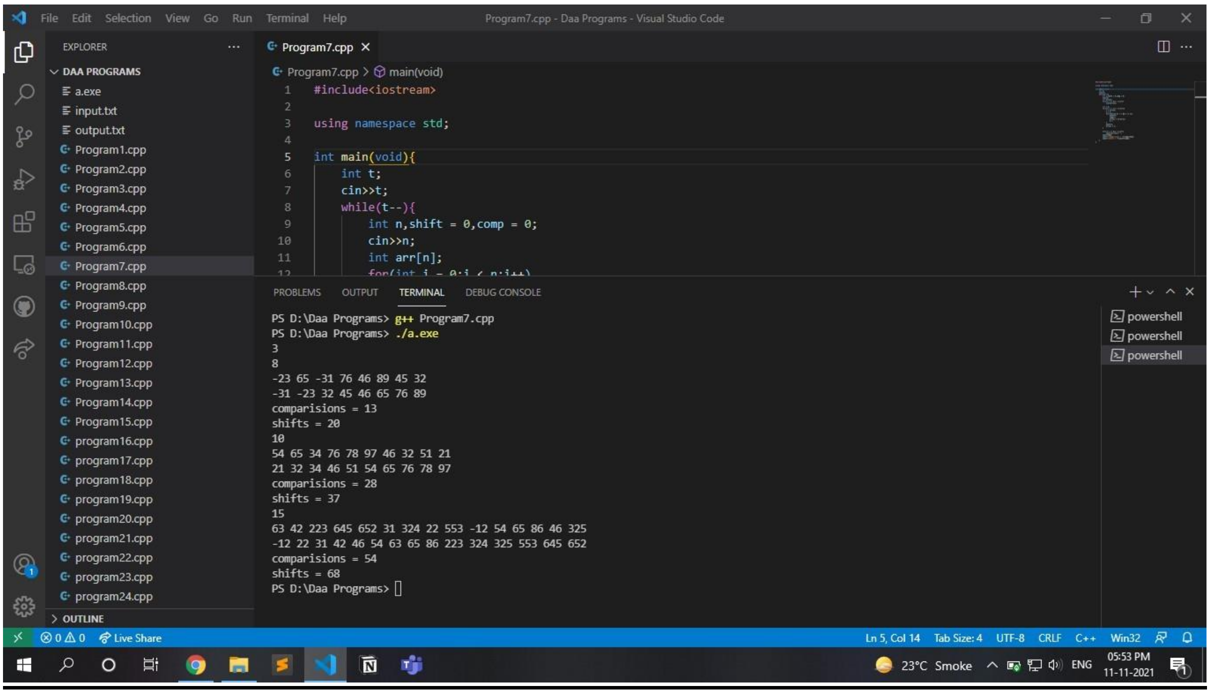
Program 7

Given an unsorted array of integers, design an algorithm and a program to sort the array using insertion sort. Your program should be able to find number of comparisons and shifts (shifts - total number of times the array elements are shifted from their place) required for sorting the array.

Code:

```
#include<iostream>

using namespace std;

int main(void){
        int t;
        cin>>t;
        while(t--){
                int n,shift = 0,comp = 0;
                cin>>n;
                int arr[n];
                for(int i = 0;i < n;i++)
                        cin>>arr[i];

                int v,j;
                for(int i = 1;i < n;i++){
                        v = arr[i];
                        j = i;
                        while(arr[j-1] > v && j >= 1){
                                shift++;
                                comp++;
                                arr[j] = arr[j-1];
                                j--;
                        }
                        shift++;
                        arr[j] = v;
                }
                for(int i = 0;i < n;i++)
                        cout<<arr[i]<<" ";
                cout<<endl;
                cout<<"comparisions = "<<comp<<endl;
                cout<<"shifts = "<<shift<<endl;
        }
        return 0;
}
```

OUTPUT



```
#include<iostream>

using namespace std;

int main(void){
    int t;
    cin>>t;
    while(t--){
        int n,shift = 0,comp = 0;
        cin>>n;
        int arr[n];
        for(int i = 0;i < n;i++)
```

```
PS D:\Daa Programs> g++ Program7.cpp
PS D:\Daa Programs> ./a.exe
3
8
-23 65 -31 76 46 89 45 32
-31 -23 32 45 46 65 76 89
comparisions = 13
shifts = 20
10
54 65 34 76 78 97 46 32 51 21
21 32 34 46 51 54 65 76 78 97
comparisions = 28
shifts = 37
15
63 42 223 645 652 31 324 22 553 -12 54 65 86 46 325
-12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
comparisions = 54
shifts = 68
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

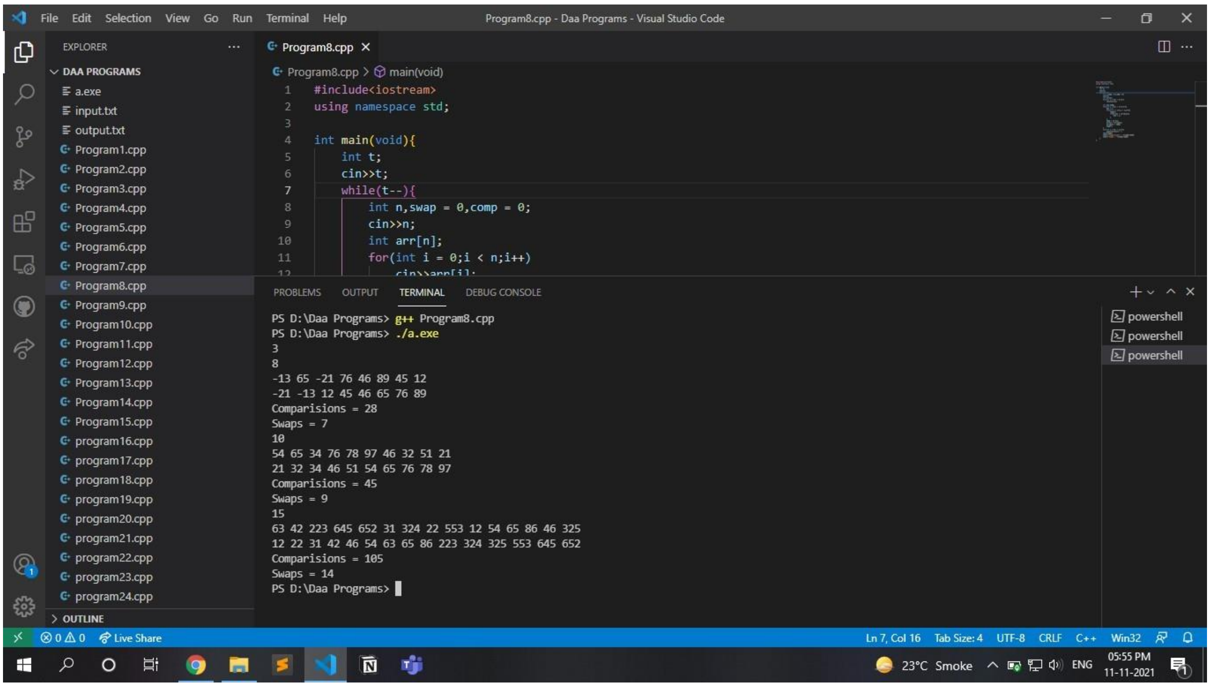
Program 8

Given an unsorted array of integers, design an algorithm and implement a program to sort this array using selection sort. Your program should also find number of comparisons and number of swaps required.

Code:

```
#include<iostream>
using namespace std;

int main(void){
	int t;
	cin>>t;
	while(t--){
		int n,swap = 0,comp = 0;
		cin>>n;
		int arr[n];
		for(int i = 0;i < n;i++)
			cin>>arr[i];

		int min,temp;
		for(int i = 0;i < n-1;i++){
			min = i;
			for(int j = i+1;j < n;j++){
				comp++;
				if(arr[j] < arr[min]){
					min = j;
				}
			}
			temp = arr[i];
			arr[i] = arr[min];
			arr[min] = temp;
			swap++;
		}
		for(int i = 0;i < n;i++)
			cout<<arr[i]<<" ";
		cout<<endl;
		cout<<"Comparisions = "<<comp<<endl;
		cout<<"Swaps = "<<swap<<endl;	}  }
```

OUTPUT



```cpp
#include<iostream>
using namespace std;

int main(void){
    int t;
    cin>>t;
    while(t--){
        int n,swap = 0,comp = 0;
        cin>>n;
        int arr[n];
        for(int i = 0;i < n;i++)
```

```
PS D:\Daa Programs> g++ Program8.cpp
PS D:\Daa Programs> ./a.exe
3
8
-13 65 -21 76 46 89 45 12
-21 -13 12 45 46 65 76 89
Comparisions = 28
Swaps = 7
10
54 65 34 76 78 97 46 32 51 21
21 32 34 46 51 54 65 76 78 97
Comparisions = 45
Swaps = 9
15
63 42 223 645 652 31 324 22 553 12 54 65 86 46 325
12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
Comparisions = 105
Swaps = 14
PS D:\Daa Programs>
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 9

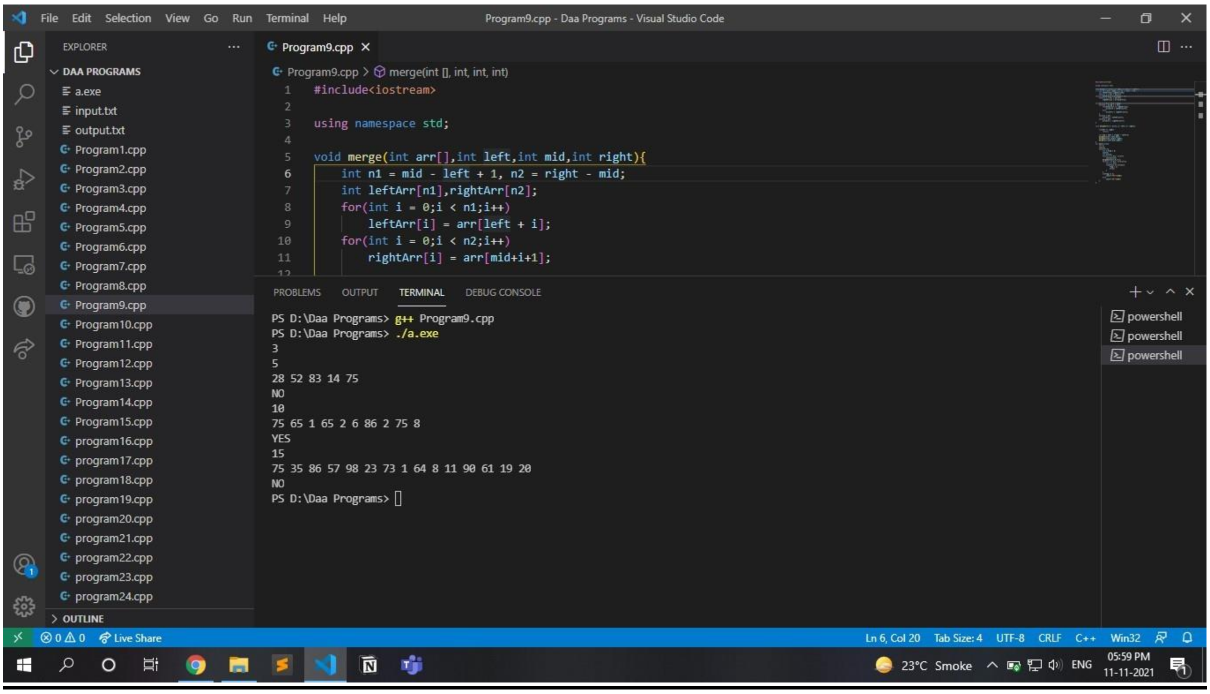
Given an unsorted array of positive integers, design an algorithm and implement it using a program to find whether there are any duplicate elements in the array or not. (use sorting) (Time Complexity = O(n log n))

Code:

```
#include<iostream>

using namespace std;

void merge(int arr[],int left,int mid,int right){
	int n1 = mid - left + 1, n2 = right - mid;
	int leftArr[n1],rightArr[n2];
	for(int i = 0;i < n1;i++)
		leftArr[i] = arr[left + i];
	for(int i = 0;i < n2;i++)
		rightArr[i] = arr[mid+i+1];

	int i = 0,j = 0,k = left;
	while(i < n1 && j < n2){
		if(leftArr[i] <= rightArr[j])
			arr[k++] = leftArr[i++];
		else
			arr[k++] = rightArr[j++];
	}
	while(i < n1)
		arr[k++] = leftArr[i++];
	while(j < n2)
		arr[k++] = rightArr[j++];
}

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

	if(left >= right)
		return;

	int mid = left + (right - left)/2;
	mergeSort(arr,left,mid);
```

```
        mergeSort(arr,mid+1,right);
        merge(arr,left,mid,right);
}
int main(void){
        int t;
        cin>>t;
        while(t--){
                int n,flag = 0;
                cin>>n;
                int arr[n];
                for(int i = 0;i < n;i++)
                        cin>>arr[i];
                mergeSort(arr,0,n-1);
                for(int i = 0;i < n-1;i++){
                        int j = i+1;
                        if(arr[i] == arr[j]){
                                flag = 1;
                                break;
                        }
                }
                if(flag == 1)
                        cout<<"YES"<<endl;
                else
                        cout<<"NO"<<endl;
        }   }
```

OUTPUT



Program9.cpp - Daa Programs - Visual Studio Code

```
#include<iostream>

using namespace std;

void merge(int arr[],int left,int mid,int right){
    int n1 = mid - left + 1, n2 = right - mid;
    int leftArr[n1],rightArr[n2];
    for(int i = 0;i < n1;i++)
        leftArr[i] = arr[left + i];
    for(int i = 0;i < n2;i++)
        rightArr[i] = arr[mid+i+1];
```

```
PS D:\Daa Programs> g++ Program9.cpp
PS D:\Daa Programs> ./a.exe
3
5
28 52 83 14 75
NO
10
75 65 1 65 2 6 86 2 75 8
YES
15
75 35 86 57 98 23 73 1 64 8 11 90 61 19 20
NO
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

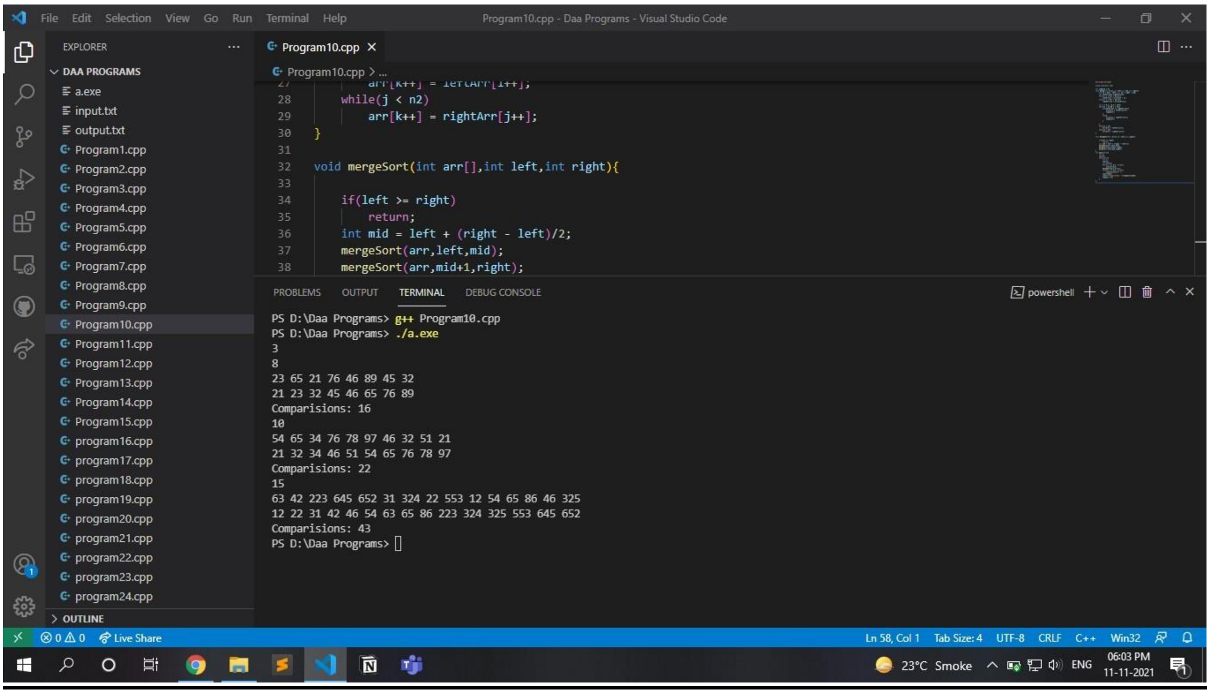
Program 10

Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by dividing the array into two subarrays and combining these subarrays after sorting each one of them. Your program should also find number of comparisons and inversions during sorting the array.

<u>Code:</u>

```
#include<iostream>

using namespace std;

int compare = 0;
void merge(int arr[],int left,int mid,int right){
	int n1 = mid - left + 1, n2 = right - mid;
	int leftArr[n1],rightArr[n2];
	for(int i = 0;i < n1;i++)
		leftArr[i] = arr[left + i];
	for(int i = 0;i < n2;i++)
		rightArr[i] = arr[mid+i+1];

	int i = 0,j = 0,k = left;
	while(i < n1 && j < n2){
		if(leftArr[i] <= rightArr[j]){
			arr[k++] = leftArr[i++];
			compare++;
		}
		else{
			arr[k++] = rightArr[j++];
			compare++;
		}

	}
	while(i < n1)
		arr[k++] = leftArr[i++];
	while(j < n2)
		arr[k++] = rightArr[j++];
}

void mergeSort(int arr[],int left,int right){
        if(left >= right)
                return;
        int mid = left + (right - left)/2;
        mergeSort(arr,left,mid);
        mergeSort(arr,mid+1,right);
        merge(arr,left,mid,right);
}
int main(void){
        int t;
        cin>>t;
        while(t--){
                int n;
                cin>>n;
                int arr[n];
                for(int i = 0;i < n;i++)
                        cin>>arr[i];
                mergeSort(arr,0,n-1);
                for(int i = 0;i < n;i++)
                        cout<<arr[i]<<" ";
                cout<<endl;
                cout<<"Comparisions: "<<compare<<endl;
                compare = 0;
        }
}
```

OUTPUT



```
        arr[k++] = leftArr[i++];
    while(j < n2)
        arr[k++] = rightArr[j++];
}

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

    if(left >= right)
        return;
    int mid = left + (right - left)/2;
    mergeSort(arr,left,mid);
    mergeSort(arr,mid+1,right);
```

```
PS D:\Daa Programs> g++ Program10.cpp
PS D:\Daa Programs> ./a.exe
3
8
23 65 21 76 46 89 45 32
21 23 32 45 46 65 76 89
Comparisions: 16
10
54 65 34 76 78 97 46 32 51 21
21 32 34 46 51 54 65 76 78 97
Comparisions: 22
15
63 42 223 645 652 31 324 22 553 12 54 65 86 46 325
12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
Comparisions: 43
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 11

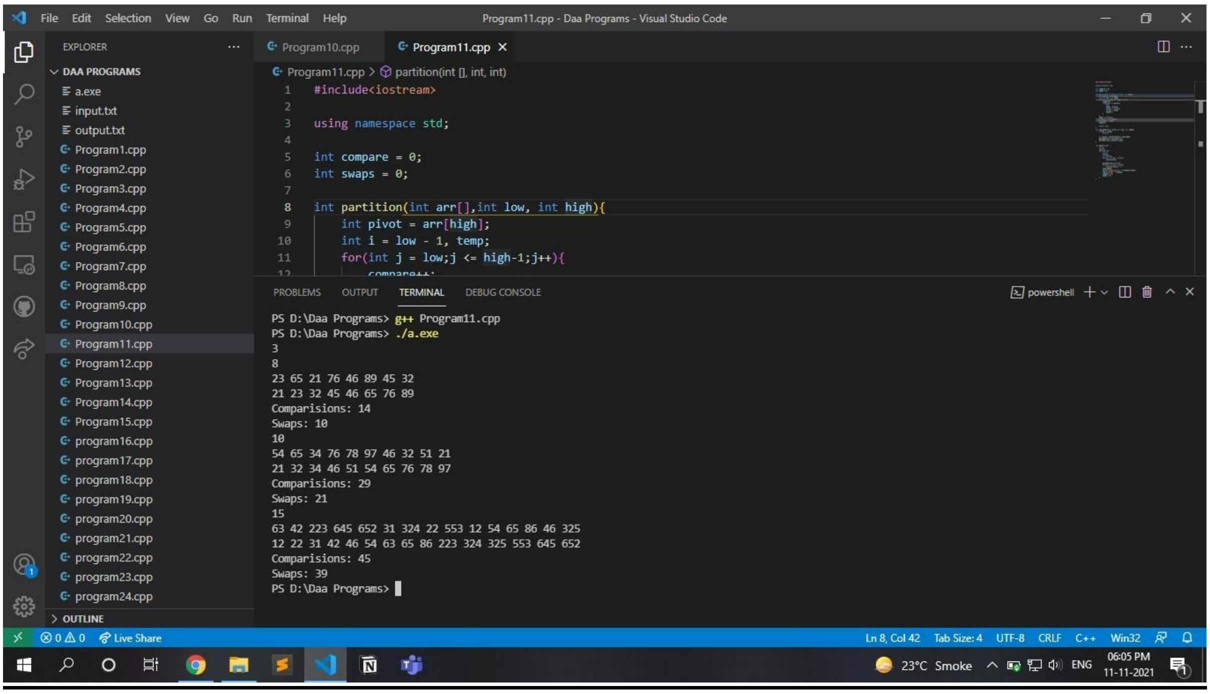
Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by partitioning the array into two subarrays based on a pivot element such that one of the sub array holds values smaller than the pivot element while another sub array holds values greater than the pivot element. Pivot element should be selected randomly from the array. Your program should also find number of comparisons and swaps required for sorting the array.

<u>Code:</u>

```
#include<iostream>

using namespace std;

int compare = 0;
int swaps = 0;

int partition(int arr[],int low, int high){
        int pivot = arr[high];
        int i = low - 1, temp;
        for(int j = low;j <= high-1;j++){
                compare++;
                if(arr[j] <= pivot){
                        i++;
                        temp = arr[i];
                        arr[i] = arr[j];
                        arr[j] = temp;
                        swaps++;
                }
        }
        temp = arr[i+1];
        arr[i+1] = arr[high];
        arr[high] = temp;
        swaps++;

        return i+1;
}
void quickSort(int arr[],int low, int high){
        if(low >= high)
```

```
                return;

        int pivot = partition(arr,low,high);
        quickSort(arr,low,pivot-1);
        quickSort(arr,pivot+1,high);
}

int main(void){
        int t;
        cin>>t;
        while(t--){
                int n;
                cin>>n;
                int arr[n];
                for(int i = 0;i < n;i++)
                        cin>>arr[i];

                quickSort(arr,0,n-1);
                for(int i = 0;i < n;i++)
                        cout<<arr[i]<<" ";
                cout<<endl;
                cout<<"Comparisions: "<<compare<<endl;
                cout<<"Swaps: "<<swaps;
                compare = 0;
                swaps = 0;
        }
}
```

OUTPUT



```
#include<iostream>

using namespace std;

int compare = 0;
int swaps = 0;

int partition(int arr[],int low, int high){
    int pivot = arr[high];
    int i = low - 1, temp;
    for(int j = low;j <= high-1;j++){
        compare++;
```

```
PS D:\Daa Programs> g++ Program11.cpp
PS D:\Daa Programs> ./a.exe
3
8
23 65 21 76 46 89 45 32
21 23 32 45 46 65 76 89
Comparisions: 14
Swaps: 10
10
54 65 34 76 78 97 46 32 51 21
21 32 34 46 51 54 65 76 78 97
Comparisions: 29
Swaps: 21
15
63 42 223 645 652 31 324 22 553 12 54 65 86 46 325
12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
Comparisions: 45
Swaps: 39
PS D:\Daa Programs>
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

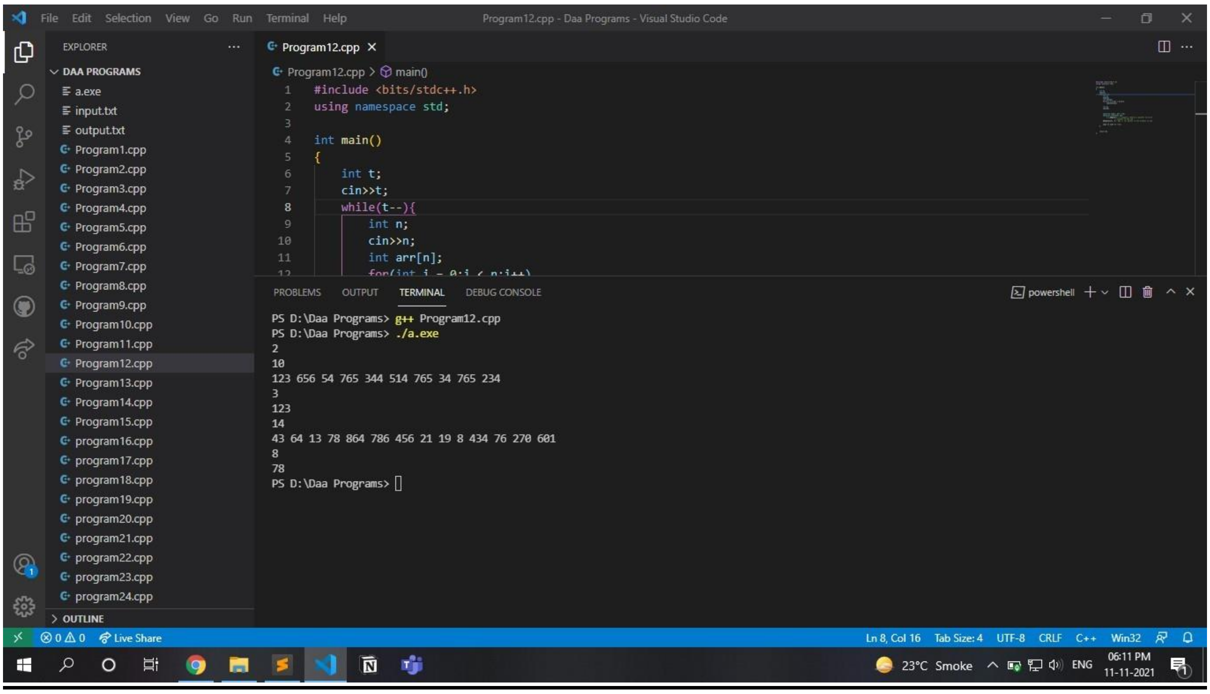
Program 12

Given an unsorted array of integers, design algorithm and implement it using a program to an find Kth smallest or largest element in the array.(Worst case Time Complexity = O(n))

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 k;
                cin>>k;


        set<int> s(arr, arr + n);
        set<int>::iterator itr
        = s.begin(); // s.begin() returns a pointer to first
              // element in the set
        advance(itr, k - 1); // itr points to kth element in set

        cout << *itr << "\n";
        }


   return 0;
}
```

OUTPUT



```cpp
#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++)
```

```
PS D:\Daa Programs> g++ Program12.cpp
PS D:\Daa Programs> ./a.exe
2
10
123 656 54 765 344 514 765 34 765 234
3
123
14
43 64 13 78 864 786 456 21 19 8 434 76 270 601
8
78
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 13

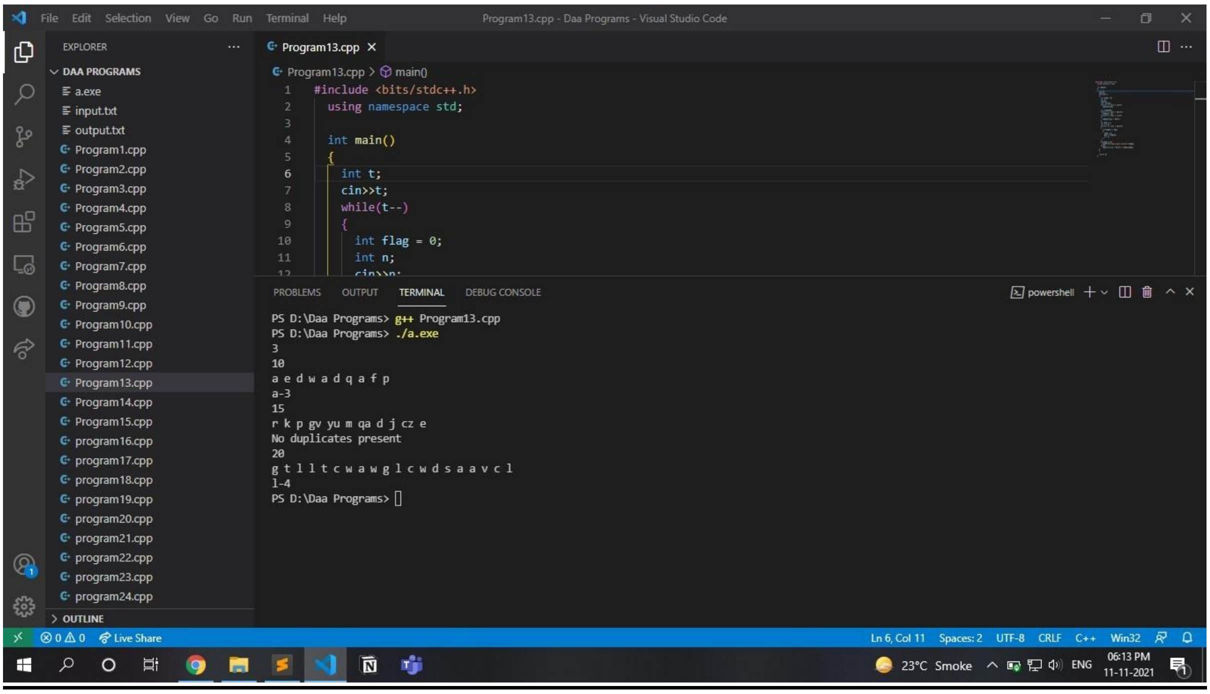
Given an unsorted array of alphabets containing duplicate elements. Design an algorithm and implement it using a program to find which alphabet has maximum number of occurrences and print it.

Code:

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

  int main()
  {
    int t;
    cin>>t;
    while(t--)
    {
      int flag = 0;
      int n;
      cin>>n;
      char arr[n];
      for(int i = 0;i < n;i++)
            cin>>arr[i];

      int freq[26];
      for(int i = 0;i < 26;i++)
        freq[i] = 0;
      for(int i = 0;i < n;i++)
      {
        freq[arr[i] - 97]++;
      }
      int max = 1;
      int j = 0;
      for(int i = 0;i < 26;i++)
      {
        if(freq[i] > max)
        {
          flag = 1;
          max = freq[i];
```

```
            j = i;
          }
        }
        if(flag == 0)
          cout<<"No duplicates present"<<endl;
        else{
              cout<<char(j + 97)<<"-"<<max<<endl;
                }
      }

      return 0;
    }
```

OUTPUT



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

  int main()
  {
    int t;
    cin>>t;
    while(t--)
    {
      int flag = 0;
      int n;
      cin>>n;
```

```
PS D:\Daa Programs> g++ Program13.cpp
PS D:\Daa Programs> ./a.exe
3
10
a e d w a d q a f p
a-3
15
r k p gv yu m qa d j cz e
No duplicates present
20
g t l l t c w a w g l c w d s a a v c l
l-4
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

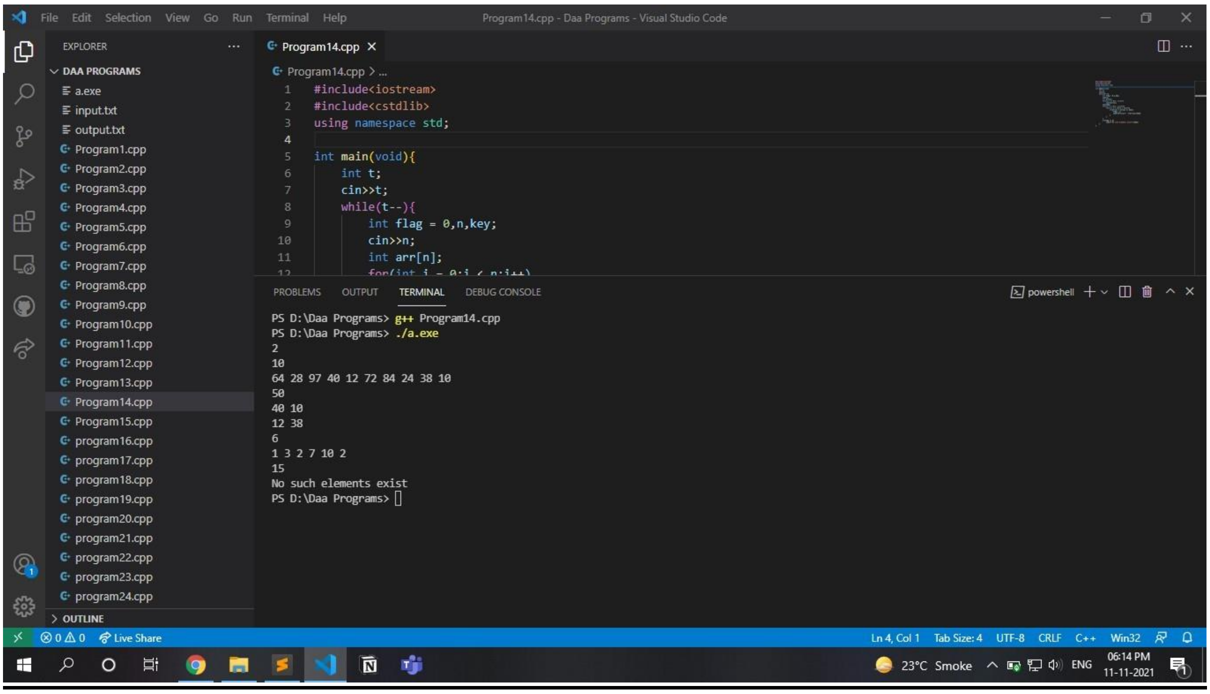
Program 14

Given an unsorted array of integers, design an algorithm and implement it using a program to find whether two elements exist such that their sum is equal to the given key element. (Time Complexity = O(n log n)).

Code:

```
#include<iostream>
#include<cstdlib>
using namespace std;

int main(void){
        int t;
        cin>>t;
        while(t--){
                int flag = 0,n,key;
                cin>>n;
                int arr[n];
                for(int i = 0;i < n;i++)
                        cin>>arr[i];
                cin>>key;
                for(int i = 0;i < n;i++){
                        for(int j = i+1;j < n;j++){
                                if(arr[i] + arr[j] == key){
                                        flag = 1;
                                        cout<<arr[i]<<" "<<arr[j]<<endl;
                                }
                        }
                }
                if(flag == 0)
                        cout<<"No such elements exist"<<endl;
        }
}
```

OUTPUT



```cpp
#include<iostream>
#include<cstdlib>
using namespace std;

int main(void){
    int t;
    cin>>t;
    while(t--){
        int flag = 0,n,key;
        cin>>n;
        int arr[n];
        for(int i = 0;i < n;i++)
```

```
PS D:\Daa Programs> g++ Program14.cpp
PS D:\Daa Programs> ./a.exe
2
10
64 28 97 40 12 72 84 24 38 10
50
40 10
12 38
6
1 3 2 7 10 2
15
No such elements exist
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 15

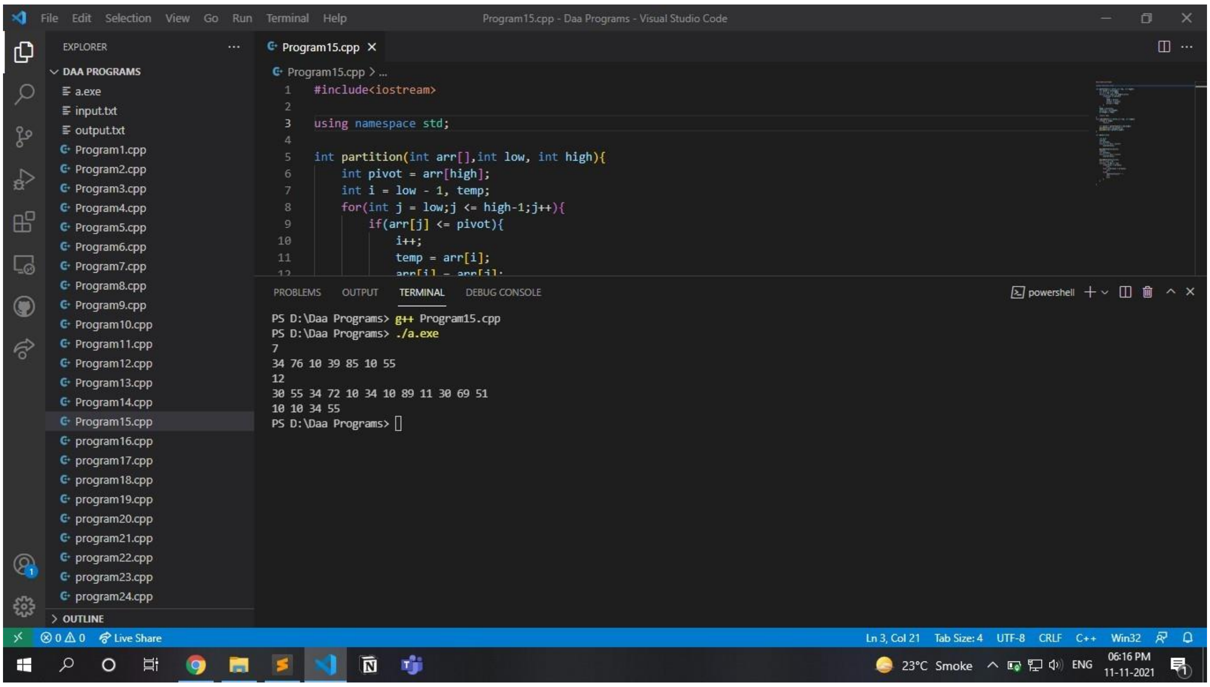
You have been given two sorted integer arrays of size m and n. Design an algorithm and implement it using a program to find list of elements which are common to both. (Time Complexity = O(m+n))

Code:

```
#include<iostream>

using namespace std;

int partition(int arr[],int low, int high){
	int pivot = arr[high];
	int i = low - 1, temp;
	for(int j = low;j <= high-1;j++){
		if(arr[j] <= pivot){
			i++;
			temp = arr[i];
			arr[i] = arr[j];
			arr[j] = temp;
		}
	}
	temp = arr[i+1];
	arr[i+1] = arr[high];
	arr[high] = temp;

	return i+1;
}
void quickSort(int arr[],int low, int high){
	if(low >= high)
		return;

	int pivot = partition(arr,low,high);
	quickSort(arr,low,pivot-1);
	quickSort(arr,pivot+1,high);
}

int main(void){

	int m,n;
```

```
        cin>>m;
        int arr1[m];
        for(int i = 0;i < m;i++)
                cin>>arr1[i];

        quickSort(arr1,0,m-1);
        cin>>n;
        int arr2[n];
        for(int i = 0;i < n;i++)
                cin>>arr2[i];

        quickSort(arr2,0,n-1);
        int i = 0,j = 0;
        while(i < m && j < n){
                if(arr1[i] > arr2[j])
                        j++;
                else if(arr1[i] < arr2[j])
                        i++;
                else{
                        cout<<arr1[i]<<" ";
                        i++;
                        j++;
                }
        }

}
```

OUTPUT



```
#include<iostream>

using namespace std;

int partition(int arr[],int low, int high){
    int pivot = arr[high];
    int i = low - 1, temp;
    for(int j = low;j <= high-1;j++){
        if(arr[j] <= pivot){
            i++;
            temp = arr[i];
            arr[i] = arr[j];
```

```
PS D:\Daa Programs> g++ Program15.cpp
PS D:\Daa Programs> ./a.exe
7
34 76 10 39 85 10 55
12
30 55 34 72 10 34 10 89 11 30 69 51
10 10 34 55
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

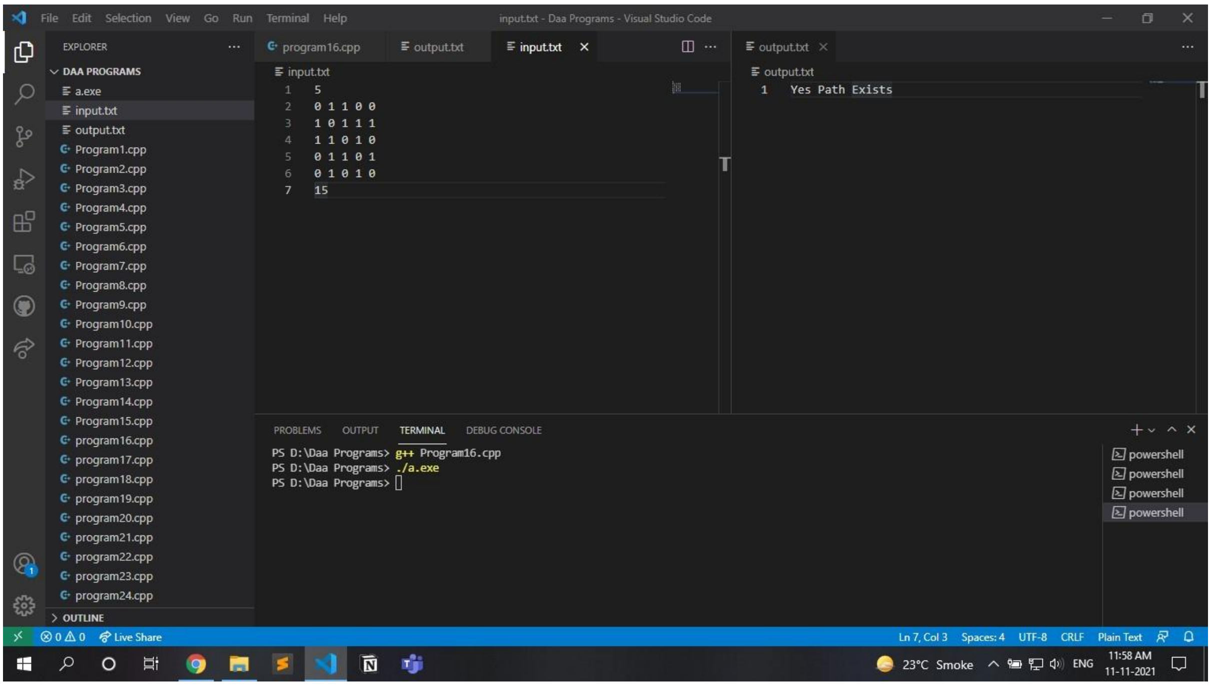
Program 16

Given a (directed/undirected) graph, design an algorithm and implement it using a program to find if a path exists between two given vertices or not. (Hint: use DFS)

Code:

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

bool findpath(int src, int dest, vector<vector<int>>& graph){
        if( src == dest)
                return true;
        int n = graph.size();
        vector<bool> visited(n, false);
        visited[src] = true;
        // DFS method
        stack<int> s;
        s.push(src);
        while(!s.empty()){
                int a = s.top();
                s.pop();
                for(int x: graph[a]){
                        if(x==dest)
                                return true;
                        if(!visited[x]){
                                visited[x] = true;
                                s.push(x);
                        }
                }
        }
        return false;
}

int main()
{

        #ifndef ONLINE_JUDGE
                freopen("input.txt","r",stdin);
                freopen("output.txt","w",stdout);
        #endif

        int n,m;
        cin>>n>>m;
        vector<vector<int>> graph(n);
        for(int i=0;i<m;i++){
                int u,v;
                cin>>u>>v;
                graph[u].push_back(v);
                graph[v].push_back(u);
        }
        int source,dest;
        cin>>source>>dest;
        if(findpath(source,dest, graph)){
                cout<<"Yes Path Exists";
        }
        else{
                cout<<"No Such Path Exists";
        }
        return 0;
}
```

OUTPUT



input.txt

```
5
0 1 1 0 0
1 0 1 1 1
1 1 0 1 0
0 1 1 0 1
0 1 0 1 0
15
```

output.txt

```
Yes Path Exists
```

```
PS D:\Daa Programs> g++ Program16.cpp
PS D:\Daa Programs> ./a.exe
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 17

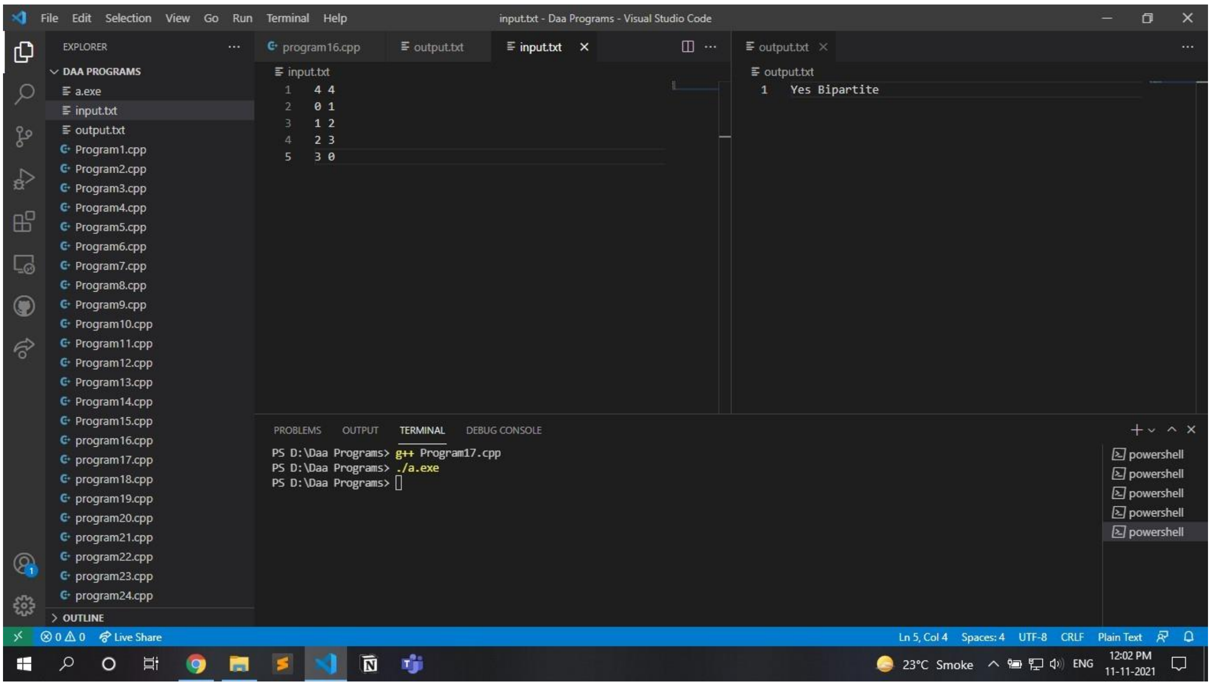
Given a graph, design an algorithm and implement it using a program to find if a graph is bipartite or not.

Code:

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

vector<vector<int>> adj;
vector<bool> vis;
vector<int> col;

bool bipart;

//assign color to nodes either 0 or 1
void color(int u, int curr){
        if(col[u] != -1 and col[u]!= curr){
                bipart = false;
                return;
        }
        col[u]= curr;
        if(vis[u])
                return;
        vis[u] = true;
        for(auto i: adj[u]){
                color(i,curr xor 1);
        }
}

int main()
{

        #ifndef ONLINE_JUDGE
                freopen("input.txt","r",stdin);
                freopen("output.txt","w",stdout);
        #endif

        int n,m;
        cin>>n>>m;
```

```
    adj = vector<vector<int>>(n);
    vis = vector<bool>(n,false);
    col = vector<int>(n,-1);
    bipart = true;
    for(int i=0;i<m;i++){
        int u,v;
        cin>>u>>v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    for(int i=0;i<n;i++){
        if(!vis[i]){
            color(i,0);
        }
    }
    if(bipart)
        cout<<"Yes Bipartite";
    else
        cout<<"Not Bipartite";

    return 0;
}
```

OUTPUT



input.txt
```
4 4
0 1
1 2
2 3
3 0
```

output.txt
```
Yes Bipartite
```

```
PS D:\Daa Programs> g++ Program17.cpp
PS D:\Daa Programs> ./a.exe
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

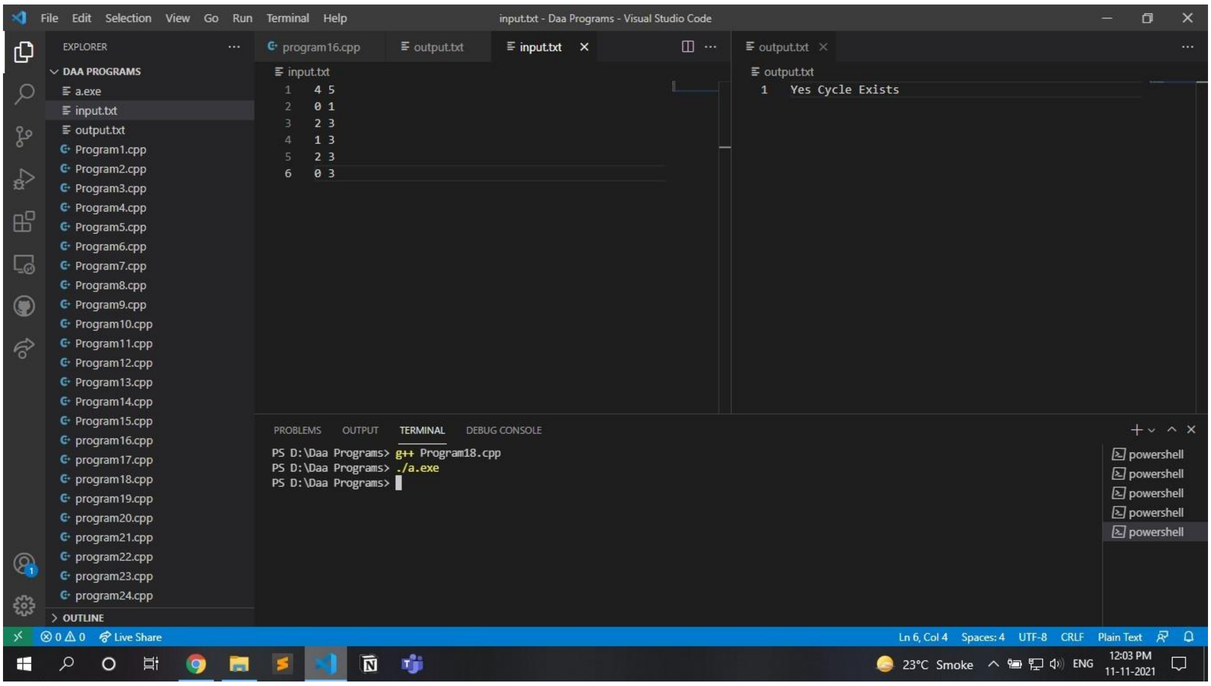
Program 18

Given a directed graph, design an algorithm and implement it using a program to find whether a cycle exists in the graph or not.

Code:

```
#include<bits/stdc++.h>
using namespace std;
bool iscycle(int src, vector<vector<int>>& adj, vector<bool>& vis, int
        parent){
        vis[src] = true;
        for(auto i: adj[src]){
                if(i != parent){
                        if(vis[i])
                                return true;
                        if(!vis[i] and iscycle(i,adj,vis,src)){
                                return true;
                        }
                }
        }
        return false;
}
int main()
{

        #ifndef ONLINE_JUDGE
                freopen("input.txt","r",stdin);
                freopen("output.txt","w",stdout);
        #endif

        int n,m;
        cin>>n>>m;
        vector<vector<int>> adj(n);
        vector<bool> vis(n,false);
        bool cycle = false;
        for(int i=0;i<m;i++){
                int u,v;
                cin>>u>>v;
                adj[u].push_back(v);
                adj[v].push_back(u);
        }
// set node 0 as visited
        vis[0] = true;
        for(int i=0;i<n;i++){
                if(!vis[i] and iscycle(i, adj, vis, -1)){
                        cycle = true;
                        break;
                }
        }
        if(cycle)
                cout<<"Yes Cycle Exists";
        else
                cout<<"Cycle Not Present";
        return 0;
}
```

OUTPUT



input.txt

```
4 5
0 1
2 3
1 3
2 3
0 3
```

output.txt

```
Yes Cycle Exists
```

```
PS D:\Daa Programs> g++ Program18.cpp
PS D:\Daa Programs> ./a.exe
PS D:\Daa Programs>
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 19

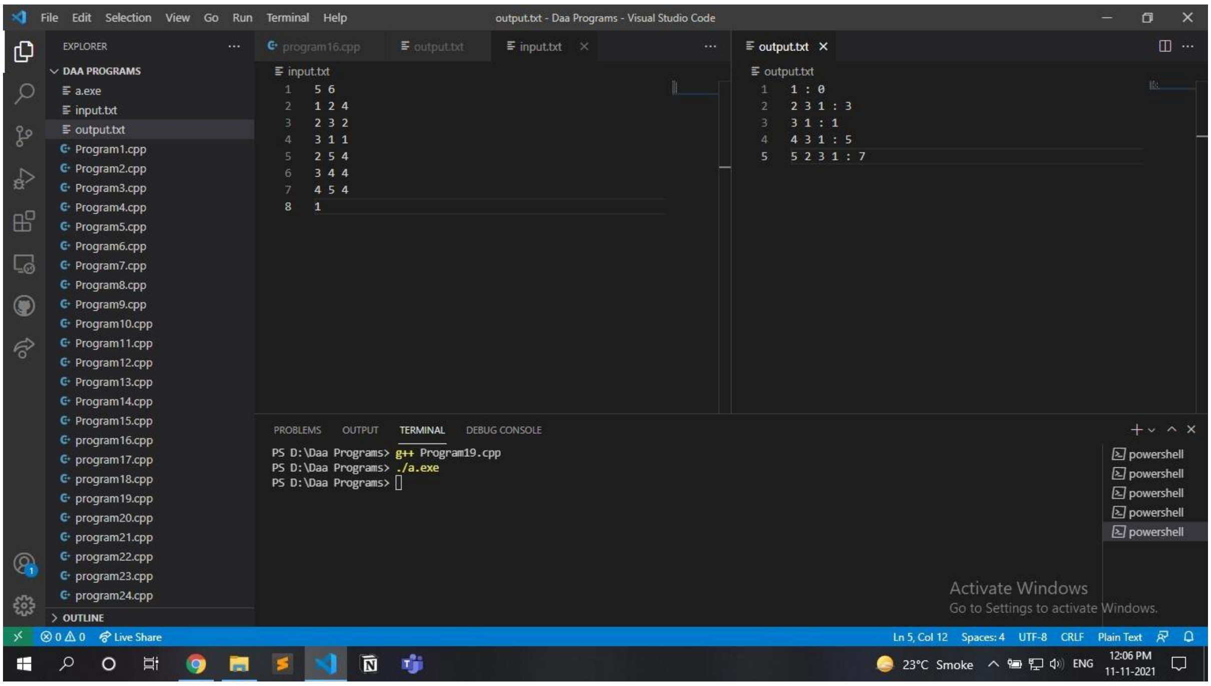
After end term examination, Akshay wants to party with his friends. All his friends are living as paying guest and it has been decided to first gather at Akshay's house and then move towards party location. The problem is that no one knows the exact address of his house in the city. Akshay as a computer science wizard knows how to apply his theory subjects in his real life and came up with an amazing idea to help his friends. He draws a graph by looking in to location of his house and his friends' location (as a node in the graph) on a map. He wishes to find out shortest distance and path covering that distance from each of his friend's location to his house and then whatsapp them this path so that they can reach his house in minimum time. Akshay has developed the program that implements Dijkstra's algorithm but not sure about correctness of results. Can you also implement the same algorithm and verify the correctness of Akshay's results? (Hint: Print shortest path and distance from friends' location to Akshay's house).

Code:

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

void path(vector<int>& parent, int j){
        if (parent[j] == - 1){
                cout<<j;
                return;
        }
        printf("%d ", j);
        path(parent, parent[j]);
}

int main()
{
        #ifndef ONLINE_JUDGE
                freopen("input.txt","r",stdin);
                freopen("output.txt","w",stdout);
        #endif

        //input graph
        int n,e;
        cin>>n>>e;
        vector<vector<pair<int,int>>> graph(n+1);
```

```
    for(int i=0;i<e;i++)
    {
        int s,d,w;
        cin>>s>>d>>w;
        graph[s].push_back({d,w});
        graph[d].push_back({s,w});
    }

    //make a distance array and a set
    vector<int> dist(n+1,INT_MAX);
    set<pair<int,int>> s;

    int source;
    cin>>source; //input source
    dist[source]=0; //initialize source distance to 0

    s.insert({0,source}); //insert source into set as {wt, vertex} pair

    vector<int> parent(n+1, -1);

    //now starts the implementation
    while(!s.empty()){
        auto x = *(s.begin()); // get the lowest weighted verte from set
        s.erase(x); //remove from set
        // now iterate through all other verteces to relax them if req
        for(auto it: graph[x.second]){
        if(dist[it.first] > dist[x.second]+it.second){ //relax
            s.erase({dist[it.first],it.first});
            dist[it.first] = dist[x.second]+it.second;
            s.insert({dist[it.first],it.first});
            parent[it.first]= x.second;
        }
        }
    }

    // print the path and shortest distance as result
    for(int i=1;i< n+1;i++){
        path(parent, i);
        cout<<" : "<<dist[i]<<endl;
    }

    return 0;
}
```

OUTPUT



input.txt

```
5 6
1 2 4
2 3 2
3 1 1
2 5 4
3 4 4
4 5 4
1
```

output.txt

```
1 : 0
2 3 1 : 3
3 1 : 1
4 3 1 : 5
5 2 3 1 : 7
```

```
PS D:\Daa Programs> g++ Program19.cpp
PS D:\Daa Programs> ./a.exe
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 20

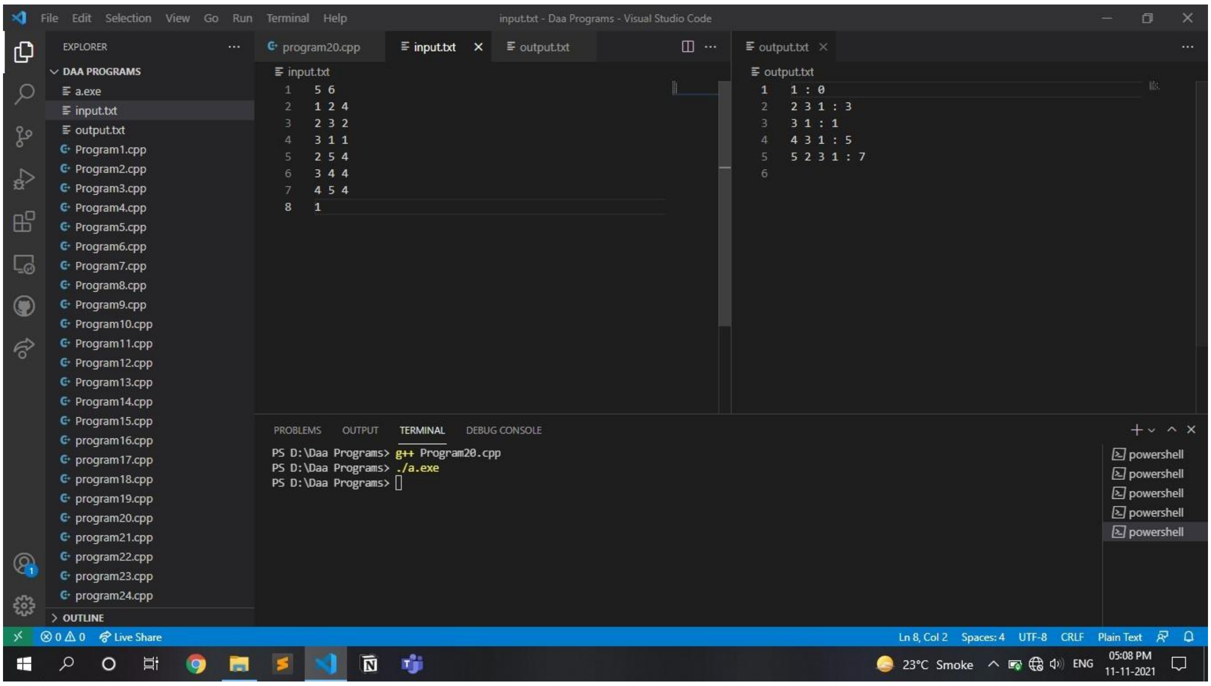
Design an algorithm and implement it using a program to solve previous question's problem using Bellman- Ford's shortest path algorithm.

Code:

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

void path(vector<int>& parent, int j){
        if (parent[j] == - 1){
                cout<<j;
                return;
        }
        printf("%d ", j);
        path(parent, parent[j]);
}

int main()
{

        #ifndef ONLINE_JUDGE
                freopen("input.txt","r",stdin);
                freopen("output.txt","w",stdout);
        #endif

        int n,e;
        cin>>n>>e;
        vector<vector<int>> edges;
        for(int i=0;i<e;i++){
                int u,v,w;
                cin>>u>>v>>w;
                edges.push_back({u,v,w});
                edges.push_back({v,u,w});
        }

        vector<int> parent(n+1,-1);
        //initialize distance array
        vector<int> dist(n+1, 1e9);
```

```
    int src;
    cin>>src; //input source
    dist[src] = 0; // initialize source distance to 0

    //iterate n-1 times to relax each edge
    bool negative_cycle;
    for(int i=1;i<n;i++)
    {

        negative_cycle = false; //to detect -ve cycle

        for(auto it: edges){
            int u,v,w;
            u = it[0];
            v = it[1];
            w = it[2];
            if(dist[v] > dist[u]+w){
                dist[v] = dist[u]+w;
                parent[v] = u;
                negative_cycle = true;
            }
        }
    }

    if(negative_cycle)
        cout<<"negative cycle present";
    else{
        for(int i=1;i< n+1;i++){
            path(parent, i);
            cout<<" : "<<dist[i]<<endl;
        }
    }

    return 0;
}
```

OUTPUT



input.txt

```
5 6
1 2 4
2 3 2
3 1 1
2 5 4
3 4 4
4 5 4
1
```

output.txt

```
1 : 0
2 3 1 : 3
3 1 : 1
4 3 1 : 5
5 2 3 1 : 7
```

```
PS D:\Daa Programs> g++ Program20.cpp
PS D:\Daa Programs> ./a.exe
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

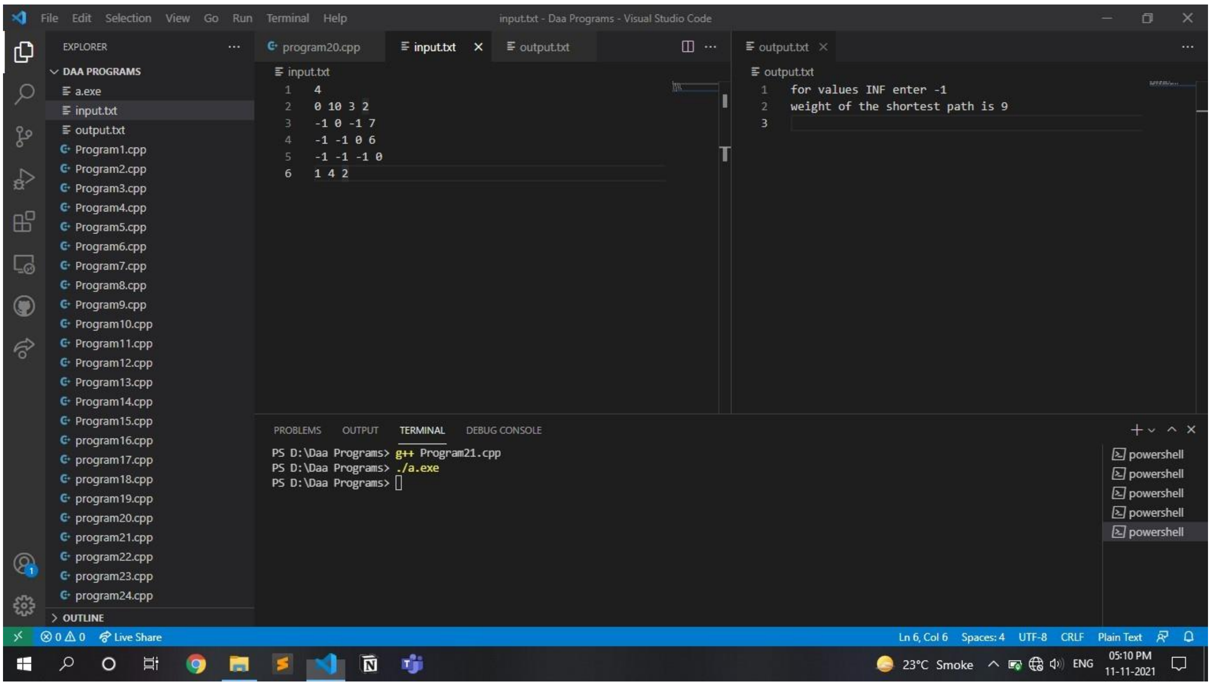
Program 21

Given a directed graph with two vertices (source and destination). Design an algorithm and implement it using a program to find the weight of the shortest path from source to destination with exactly k edges on the path.

Code:

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

#define V 100
#define INF INT_MAX
int arr[100][100];

int shortestpath(int arr[][V],int u,int v,int k, int n)
{
	if(k==0 && u==v) return 0;
	if(k==1 && arr[u][v] != INF) return arr[u][v];
	if(k<=0)
		return INF;
	int res = INF;
	for(int i=0;i<n;i++)
	{
		if(arr[u][i] != INF && u!=i && v!=i)
		{
			int rec_res = shortestpath(arr, i,v,k-1, n);
			if(rec_res != INF)
				res = min(res, arr[u][i]+rec_res);
		}
	}
	return res;
}

int main()
{

	#ifndef ONLINE_JUDGE
		freopen("input.txt","r",stdin);
		freopen("output.txt","w",stdout);
	#endif

	int n;
	cout<<"for values INF enter -1"<<endl;
	cin>>n;
	int a;

	for(int i=0;i<n;i++){
		for(int j=0;j<n;j++){
			cin>>a;
			if(a<0){
				arr[i][j] = INF;
			}
			else
				arr[i][j] = a;
		}
	}
	int u,v,k;
	cin>>u>>v>>k;

	cout<<"weight of the shortest path is "<<shortestpath(arr,u-1,v-1,k,n)<<endl;// 0 indexing is followed
	return 0;
}
```

OUTPUT



input.txt

```
4
0 10 3 2
-1 0 -1 7
-1 -1 0 6
-1 -1 -1 0
1 4 2
```

output.txt

```
for values INF enter -1
weight of the shortest path is 9
```

```
PS D:\Daa Programs> g++ Program21.cpp
PS D:\Daa Programs> ./a.exe
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

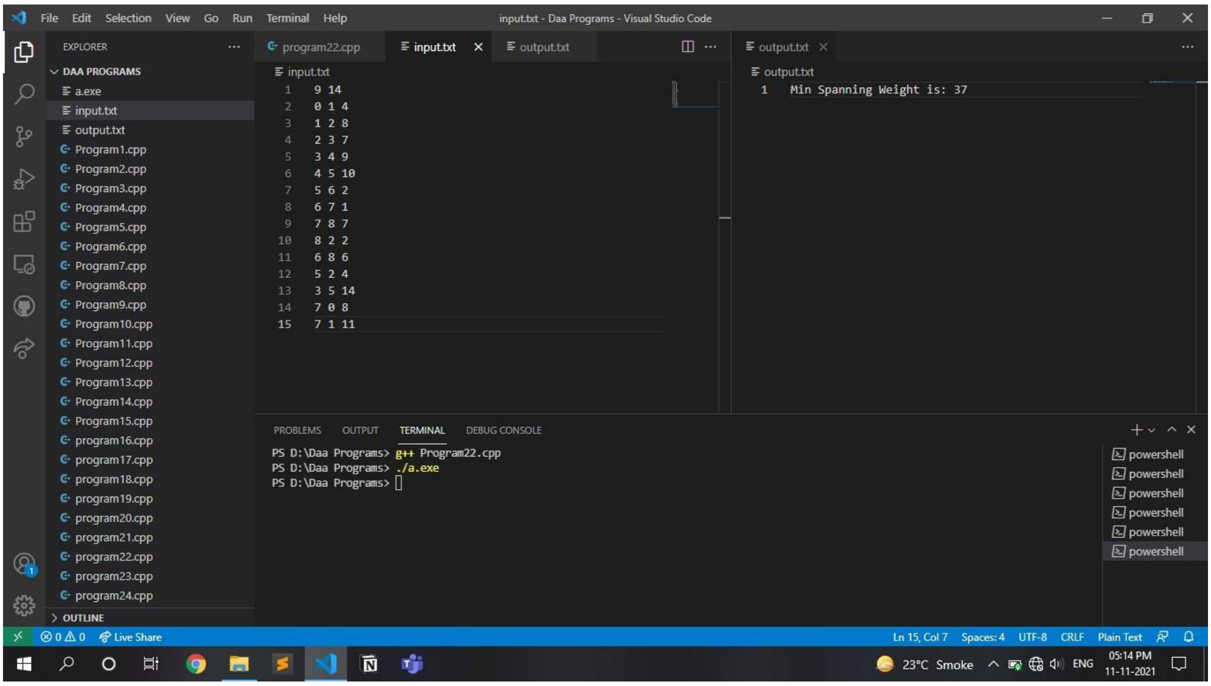
Program 22

Assume that a project of road construction to connect some cities is given to your friend. Map of these cities and roads which will connect them (after construction) is provided to him in the form of a graph. Certain amount of rupees is associated with construction of each road. Your friend has to calculate the minimum budget required for this project. The budget should be designed in such a way that the cost of connecting the cities should be minimum and number of roads required to connect all the cities should be minimum (if there are N cities then only N-1 roads need to be constructed). He asks you for help. Now, you have to help your friend by designing an algorithm which will find minimum cost required to connect these cities. (use Prim's algorithm)

Code:

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

int main()
{

        #ifndef ONLINE_JUDGE
                freopen("input.txt","r",stdin);
                freopen("output.txt","w",stdout);
        #endif

        int nodes, edges;
        cin>>nodes;
        cin>>edges;

        vector<pair<int,int>> graph[nodes];
        int source, destination, weight;

        for(int i=0;i<edges;i++)
        {
                cin>>source>>destination>>weight;
                graph[source].push_back(make_pair(destination,weight));
                graph[destination].push_back(make_pair(source, weight));
        }

        int key[nodes];//to select the min weight
        int parent[nodes];// to store the parent node
        bool mst[nodes]; // to construct the path

        for(int i=0;i< nodes;i++)// initialization
        {
                key[i]= INT_MAX;
                mst[i] = false;
                parent[i] = -1;
        }

        // priority queue APPROACH
        priority_queue<pair<int, int>, vector<pair<int,int>>,
        greater<pair<int,int>>> pq;

        key[0] = 0;//select a node to start from
        parent[0] = 0;

        pq.push({0, 0}); //{weight, index of starting node}

        for(int i=0;i< nodes-1;i++)
        {
                int u = pq.top().second;//get the index of top node
                pq.pop();//remove the node from queue
                mst[u] = true;//set mst as true for the node u

                for(auto it: graph[i])
                {
                        int dest = it.first;
                        int wt = it.second;
                        if(mst[dest]== false && wt<key[dest])//check if the parent
                                array needs to be changed
                        {
                                parent[dest] = u;
                                pq.push({key[dest], dest});
                                key[dest] = wt;
                        }
                }
        }

        int mstwt = 0;
        // to print the list with minimun weight

        for(int i=0;i<nodes;i++)
                mstwt += key[i];

        cout<<"Min Spanning Weight is: "<<mstwt;

        return 0;    }
```

OUTPUT



input.txt

```
9 14
0 1 4
1 2 8
2 3 7
3 4 9
4 5 10
5 6 2
6 7 1
7 8 7
8 2 2
6 8 6
5 2 4
3 5 14
7 0 8
7 1 11
```

output.txt

```
1   Min Spanning Weight is: 37
```

```
PS D:\Daa Programs> g++ Program22.cpp
PS D:\Daa Programs> ./a.exe
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 23

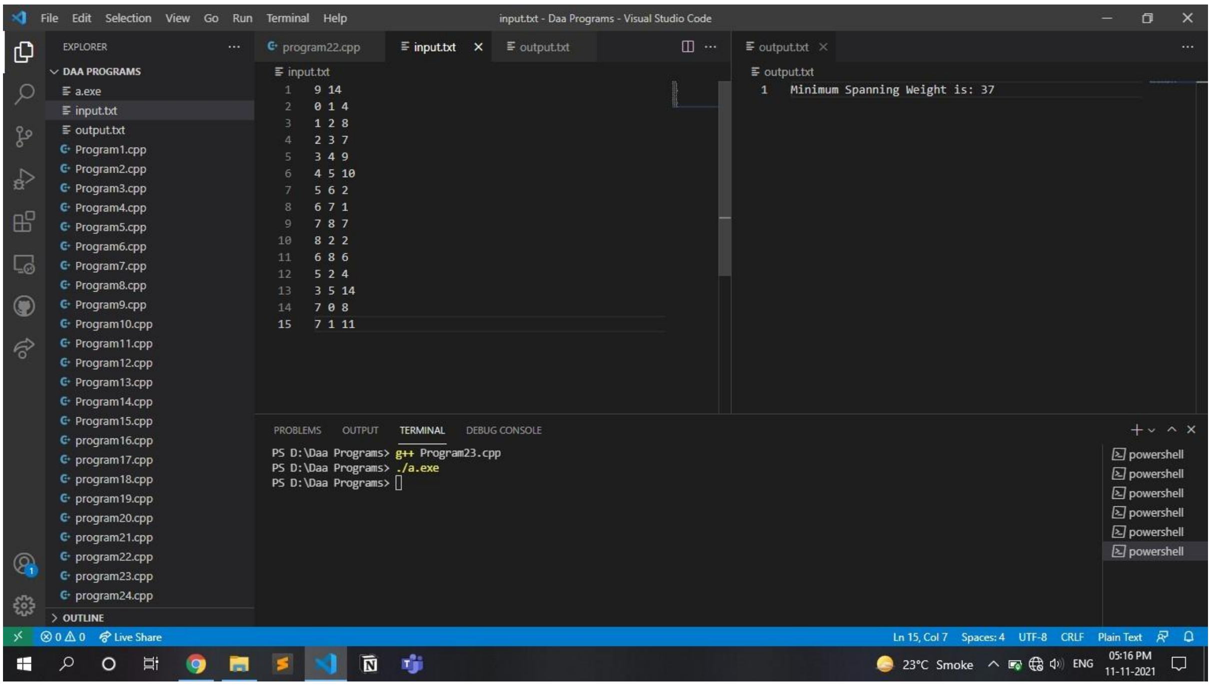
Implement the previous problem using Kruskal's algorithm.

Code:

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

vector<int> parent(100);
vector<int> sz(100);

void make_set(int v)
{
	parent[v]=v;
	sz[v] = 1;
}

int find_set(int v)
{
	if(v==parent[v])
		return v;
	return parent[v] = find_set(parent[v]);
}

void union_set(int a, int b)
{
	a = find_set(a);
	b = find_set(b);
	if(a != b){//dont belong to same set
		if(sz[a] < sz[b])
			swap(a,b);
		parent[b] = a;
		sz[a] += sz[b];
	}
}

int main()
{

	#ifndef ONLINE_JUDGE
```

```
        freopen("input.txt","r",stdin);
        freopen("output.txt","w",stdout);
    #endif

    int n,e;
    cin>>n>>e;

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

    vector<vector<int>> graph;

    for(int i=0;i<e;i++)
    {
        int u,v,w;
        cin>>u>>v>>w;
        graph.push_back({w,u,v});
        graph.push_back({w,v,u});
    }

    sort(graph.begin(), graph.end());//sort according to weight

    int total_weight = 0;

    for(auto i: graph){
        int w = i[0];
        int u = i[1];
        int v = i[2];
        int x = find_set(u);
        int y = find_set(v);
        if(x == y){
            continue;
        }
        else{
            total_weight += w;
        union_set(u,v);//add to set
        }
    }

    cout<<"Minimum Spanning Weight is: "<<total_weight;
    return 0;
}
```

OUTPUT



input.txt

```
9 14
0 1 4
1 2 8
2 3 7
3 4 9
4 5 10
5 6 2
6 7 1
7 8 7
8 2 2
6 8 6
5 2 4
3 5 14
7 0 8
7 1 11
```

output.txt

```
Minimum Spanning Weight is: 37
```

```
PS D:\Daa Programs> g++ Program23.cpp
PS D:\Daa Programs> ./a.exe
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 24

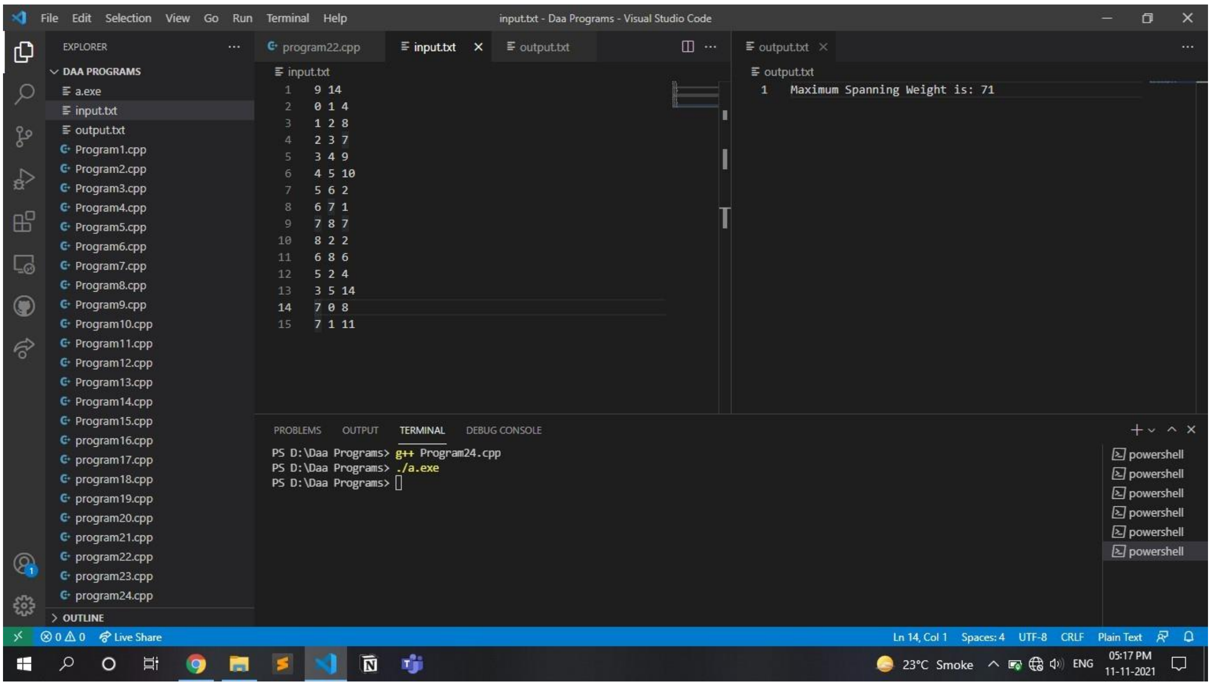
Assume that the same road construction project is given to another person. The amount he will earn from this project is directly proportional to the budget of the project. This person is greedy, so he decided to maximize the budget by constructing those roads which have the highest construction cost. Design an algorithm and implement it using a program to find the maximum budget required for the project.

Code:

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

bool compare(const pair<int,int>& a, const pair<int, int>& b){
        return b.first > a.first;
}

vector<int> parent(100);
vector<int> sz(100);

void make_set(int v)
{
        parent[v]=v;
        sz[v] = 1;
}

int find_set(int v)
{
        if(v==parent[v])
                return v;
        return parent[v] = find_set(parent[v]);
}

void union_set(int a, int b)
{
        a = find_set(a);
        b = find_set(b);
        if(a != b){//dont belong to same set
                if(sz[a] < sz[b])
                        swap(a,b);
                parent[b] = a;
        sz[a] += sz[b];
    }
}

int main()
{

    #ifndef ONLINE_JUDGE
        freopen("input.txt","r",stdin);
        freopen("output.txt","w",stdout);
    #endif

    int n,e;
    cin>>n>>e;
    for(int i=0;i<n;i++)
        make_set(i);

    vector<vector<int>> graph;

    for(int i=0;i<e;i++)
    {
        int u,v,w;
        cin>>u>>v>>w;
        graph.push_back({w,u,v});
        graph.push_back({w,v,u});
    }

    sort(graph.rbegin(), graph.rend());//sort according to weight

    int total_weight = 0;

    for(auto i: graph){
        int w = i[0];
        int u = i[1];
        int v = i[2];
        int x = find_set(u);
        int y = find_set(v);
        if(x == y){
            continue;
        }

        else{
        //cout<<u<<" "<<v<<endl;
                total_weight += w;
        union_set(u,v);//add to set
        }
    }
```

```
        cout<<"Maximum Spanning Weight is: "<<total_weight;
        return 0;
}
```

OUTPUT



input.txt

```
9 14
0 1 4
1 2 8
2 3 7
3 4 9
4 5 10
5 6 2
6 7 1
7 8 7
8 2 2
6 8 6
5 2 4
3 5 14
7 0 8
7 1 11
```

output.txt

```
Maximum Spanning Weight is: 71
```

```
PS D:\Daa Programs> g++ Program24.cpp
PS D:\Daa Programs> ./a.exe
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 25

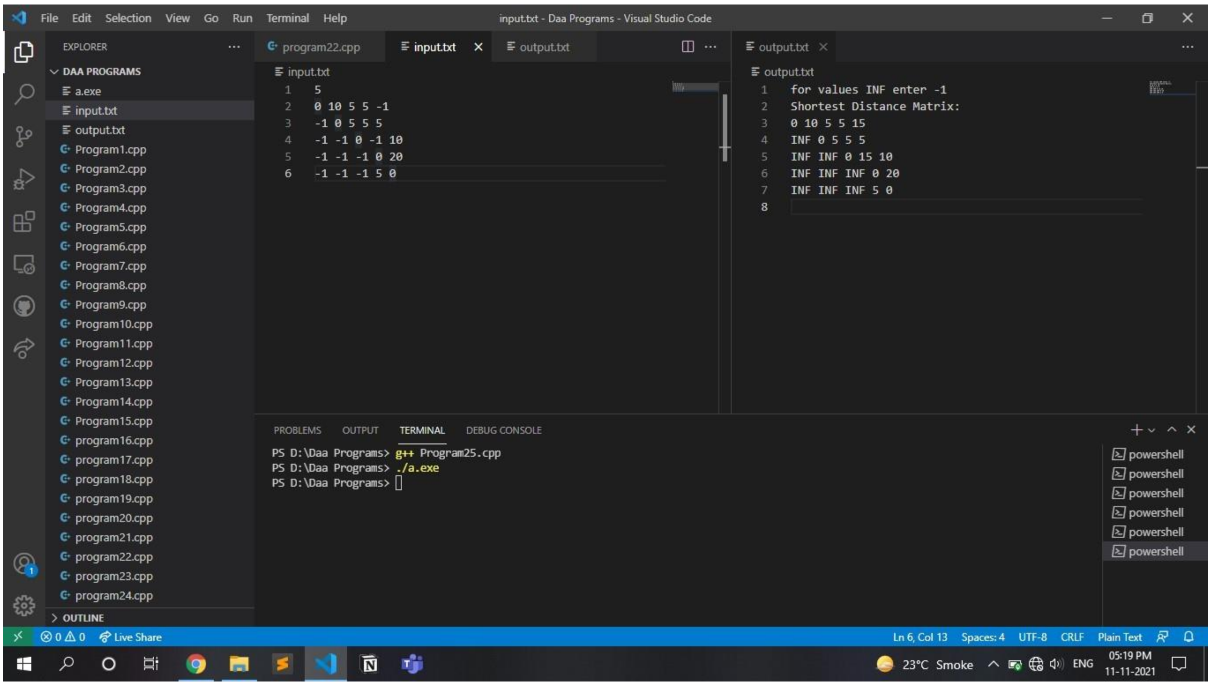
Given a graph, Design an algorithm and implement it using a program to implement FloydWarshall all pair shortest path algorithm

Code:

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

#define INF 1e9

int main()
{

	#ifndef ONLINE_JUDGE
		freopen("input.txt","r",stdin);
		freopen("output.txt","w",stdout);
	#endif

	int n;
	cout<<"for values INF enter -1"<<endl;
	cin>>n;
	int a;

	int arr[n][n], dist[n][n];

	for(int i=0;i<n;i++){
		for(int j=0;j<n;j++){
			cin>>a;
			if(a<0){
				arr[i][j] = INF;
			}
			else
				arr[i][j] = a;
			dist[i][j] = arr[i][j];
		}
	}

	for(int k=0;k<n;k++){
		for(int i=0;i<n;i++){
```

```
                for(int j=0;j<n;j++){
                    if(dist[i][k]+dist[k][j] < dist[i][j]){
                        dist[i][j] = dist[i][k]+dist[k][j];
                    }
                }
            }
        }

    cout<<"Shortest Distance Matrix: "<<endl;

    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            if(dist[i][j]==INF){
                cout<<"INF ";
            }
            else
                cout<<dist[i][j]<<" ";
        }
        cout<<endl;
    }

    return 0;
}
```

OUTPUT



input.txt

```
5
0 10 5 5 -1
-1 0 5 5 5
-1 -1 0 -1 10
-1 -1 -1 0 20
-1 -1 -1 5 0
```

output.txt

```
for values INF enter -1
Shortest Distance Matrix:
0 10 5 5 15
INF 0 5 5 5
INF INF 0 15 10
INF INF INF 0 20
INF INF INF 5 0
```

```
PS D:\Daa Programs> g++ Program25.cpp
PS D:\Daa Programs> ./a.exe
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 26

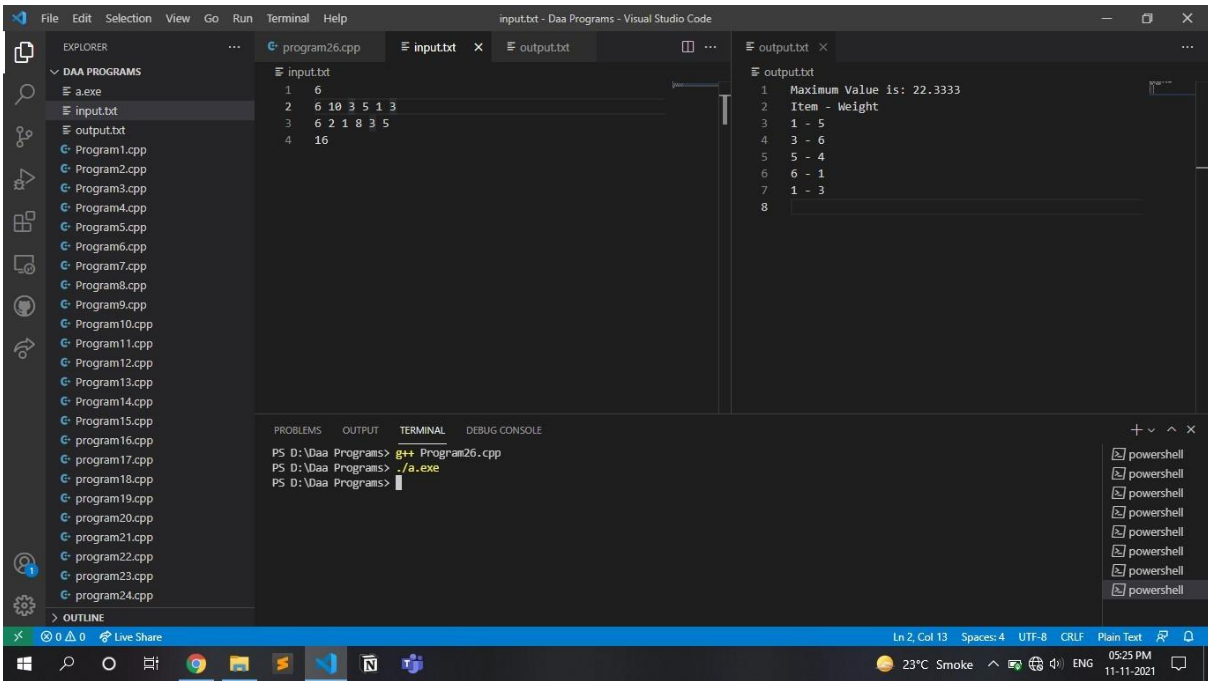
Given a knapsack of maximum capacity w. N items are provided, each having its own value and weight. You have to Design an algorithm and implement it using a program to find the list of the selected items such that the final selected content has weight w and has maximum value. You can take fractions of items,i.e. the items can be broken into smaller pieces so that you have to carry only a fraction xi of item i, where 0 ≤xi≤ 1.

<u>Code:</u>

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

int main()
{

	#ifndef ONLINE_JUDGE
		freopen("input.txt","r",stdin);
		freopen("output.txt","w",stdout);
	#endif

	int n;
	cin>>n;

	vector<double> items(n);
	vector<double> val(n);
	vector<vector<double>> job;//to store pair of

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

	for(int i=0;i<n;i++){
		cin>>val[i];
		job.push_back({val[i]/items[i],items[i],i+1});
	}
```

```
    double k;
    cin>>k;

    sort(job.rbegin(), job.rend());//sort acc to val per wt

    vector<pair<double,double>> ls;

    float profit =0;

    for(int i=0;i<n;i++)
    {
        if(job[i][1] >= k)
        {
            profit += k*job[i][0];
            ls.push_back(make_pair(k, job[i][2]));
            break;
        }
        else
        {
            profit += job[i][1]*job[i][0];
        }
        ls.push_back(make_pair(job[i][1], job[i][2]));
        k = k - job[i][1];
    }

    cout<<"Maximum Value is: "<<profit<<endl;
    cout<<"Item - Weight"<<endl;

    for(auto it: ls)
        cout<<it.first<<" - "<<it.second<<endl;

    return 0;
}
```

OUTPUT



input.txt

```
6
6 10 3 5 1 3
6 2 1 8 3 5
16
```

output.txt

```
Maximum Value is: 22.3333
Item - Weight
1 - 5
3 - 6
5 - 4
6 - 1
1 - 3

```

```
PS D:\Daa Programs> g++ Program26.cpp
PS D:\Daa Programs> ./a.exe
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

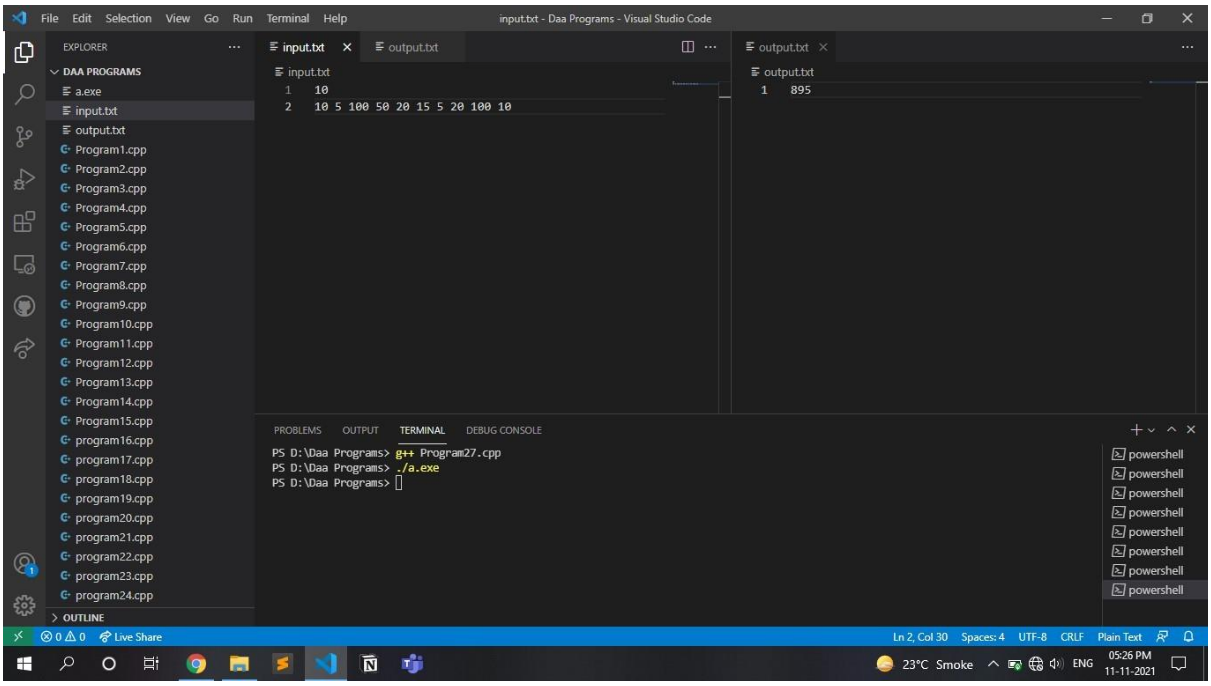
Program 27

Given an array of elements. Assume arr[i] represents the size of file i. Write an algorithm and a program to merge all these files into single file with minimum computation. For given two files A and B with sizes m and n, computation cost of merging them is O(m+n). (Hint: use greedy approach)

Code:

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

int main()
{

        #ifndef ONLINE_JUDGE
                freopen("input.txt","r",stdin);
                freopen("output.txt","w",stdout);
        #endif

        int n;
        cin>>n;

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

        priority_queue<int, vector<int>, greater<int>> minheap;

        for(int i=0;i<n;i++){
                minheap.push(a[i]);
        }

        int ans = 0;

        while(minheap.size()>1){
                int e1 = minheap.top();
                minheap.pop();
                int e2 = minheap.top();
                minheap.pop();
                ans += e1+e2;
                minheap.push(e1 + e2);
        }
        cout<<ans;
        return 0;
}
```

OUTPUT



input.txt

```
10
10 5 100 50 20 15 5 20 100 10
```

output.txt

```
895
```

```
PS D:\Daa Programs> g++ Program27.cpp
PS D:\Daa Programs> ./a.exe
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 28

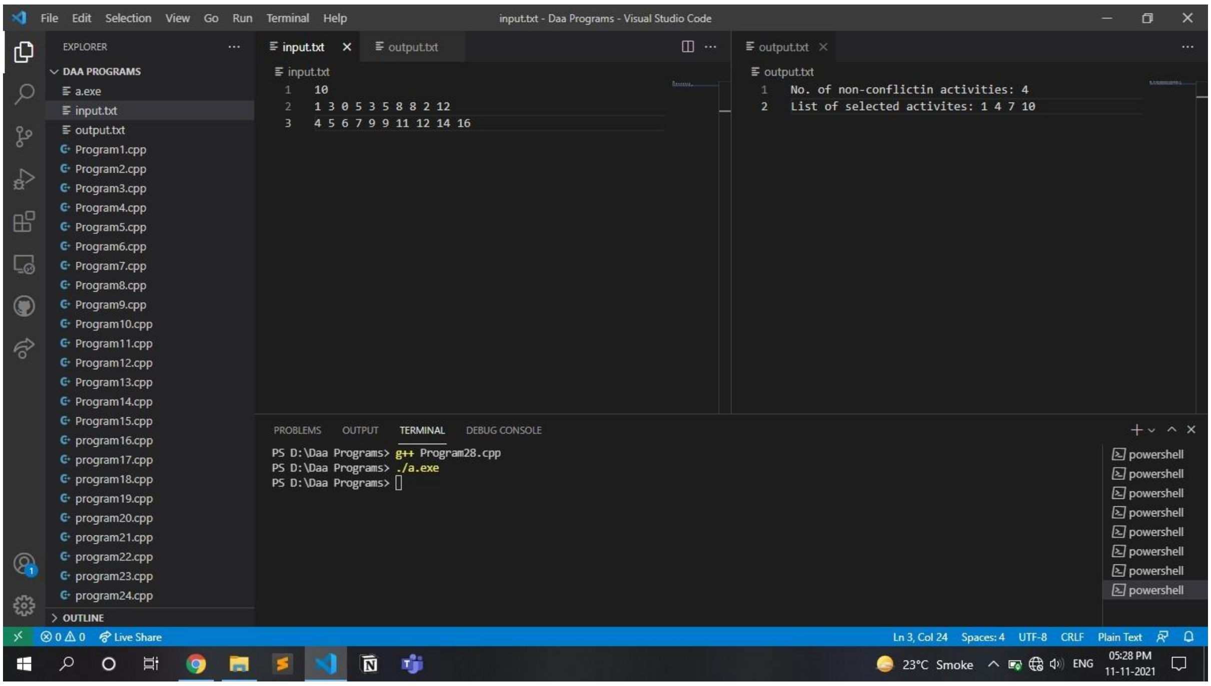
Given a list of activities with their starting time and finishing time. Your goal is to select maximum number of activities that can be performed by a single person such that selected activities must be non-conflicting. Any activity is said to be non-conflicting if starting time of an activity is greater 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.

Code:

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

int main()
{

	#ifndef ONLINE_JUDGE
		freopen("input.txt","r",stdin);
		freopen("output.txt","w",stdout);
	#endif

	int n;
	cin>>n;

	vector<int> st(n), dline(n);
	vector<vector<int>> activity;

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

	for(int i=0;i<n;i++){
		cin>>dline[i];
		activity.push_back({dline[i],st[i], i+1});
	}

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

	vector<int> selected;
	int count=0;
	int currentEnd = -1;
```

```
    for(int i=0;i<n;i++){
        if(activity[i][1]>currentEnd){
            count++;
            currentEnd = activity[i][0];
            selected.push_back(activity[i][2]);
        }
    }

    cout<<"No. of non-conflictin activities: "<<count<<endl;
    cout<<"List of selected activites: ";

    for(auto i: selected){
        cout<<i<<" ";
    }
    return 0;
}
```

OUTPUT



input.txt - Daa Programs - Visual Studio Code

input.txt

```
10
1 3 0 5 3 5 8 8 2 12
4 5 6 7 9 9 11 12 14 16
```

output.txt

```
No. of non-conflictin activities: 4
List of selected activites: 1 4 7 10
```

```
PS D:\Daa Programs> g++ Program28.cpp
PS D:\Daa Programs> ./a.exe
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 29

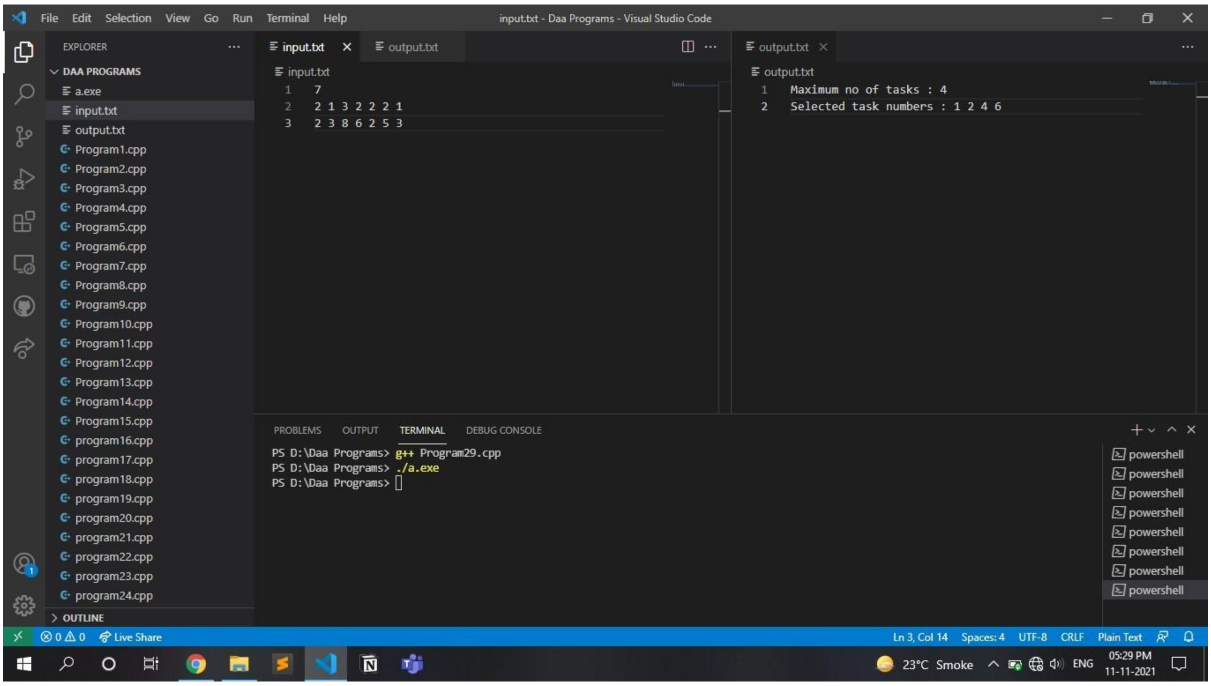
Given a long list of tasks. Each task take specific time to accomplish it and each task has a deadline associated with it. You have to design an algorithm and implement it using a program to find maximum number of tasks that can be completed without crossing their deadlines and also find list of selected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

Code:

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

bool compare(pair<int,int>a,pair<int,int>b){
        return a.first > b.first;
}

int main()
{

        #ifndef ONLINE_JUDGE
                freopen("input.txt","r",stdin);
                freopen("output.txt","w",stdout);
        #endif

        int n;
        cin>>n;

        int p[n];
        int d[n];

        for(int i=0;i<n;i++)
                cin>>p[i];

        for(int i=0;i<n;i++)
                cin>>d[i];

        vector<pair<int,int> > jobs;
```

```
    int profit,deadline;

    for(int i=0;i<n;i++){
        jobs.push_back(make_pair(p[i],d[i]));
    }

    sort(jobs.begin(),jobs.end(),compare);
    int maxEndTime = 0;

    for(int i=0;i<n;i++){
        if(jobs[i].second > maxEndTime)
            maxEndTime = jobs[i].second;
    }

    vector<int> ans;
    int fill[maxEndTime];
    int count = 0, maxProfit = 0;

    for(int i=0;i<n;i++) fill[i] = -1;
        for(int i=0;i<n;i++){
            int j = jobs[i].second - 1;
            while(j>=0 && fill[i]!=-1) j--;
            if(j>=0 && fill[j]==-1){
                fill[j] = i;
                ans.push_back(i);
                count++;
                maxProfit = maxProfit + jobs[i].first;
            }
        }

    cout<<"Maximum no of tasks : "<<count<<endl;
    cout<<"Selected task numbers : ";

    for(int i=0;i<ans.size();i++)
        cout<<ans[i]<<" ";

    return 0;
}
```

OUTPUT



input.txt

```
7
2 1 3 2 2 2 1
2 3 8 6 2 5 3
```

output.txt

```
Maximum no of tasks : 4
Selected task numbers : 1 2 4 6
```

```
PS D:\Daa Programs> g++ Program29.cpp
PS D:\Daa Programs> ./a.exe
PS D:\Daa Programs> 
```

Name: Madhav Singh
Section: I
Class Roll No: 35
Student ID: 190111071
Course: BTECH (CSE)

Program 30

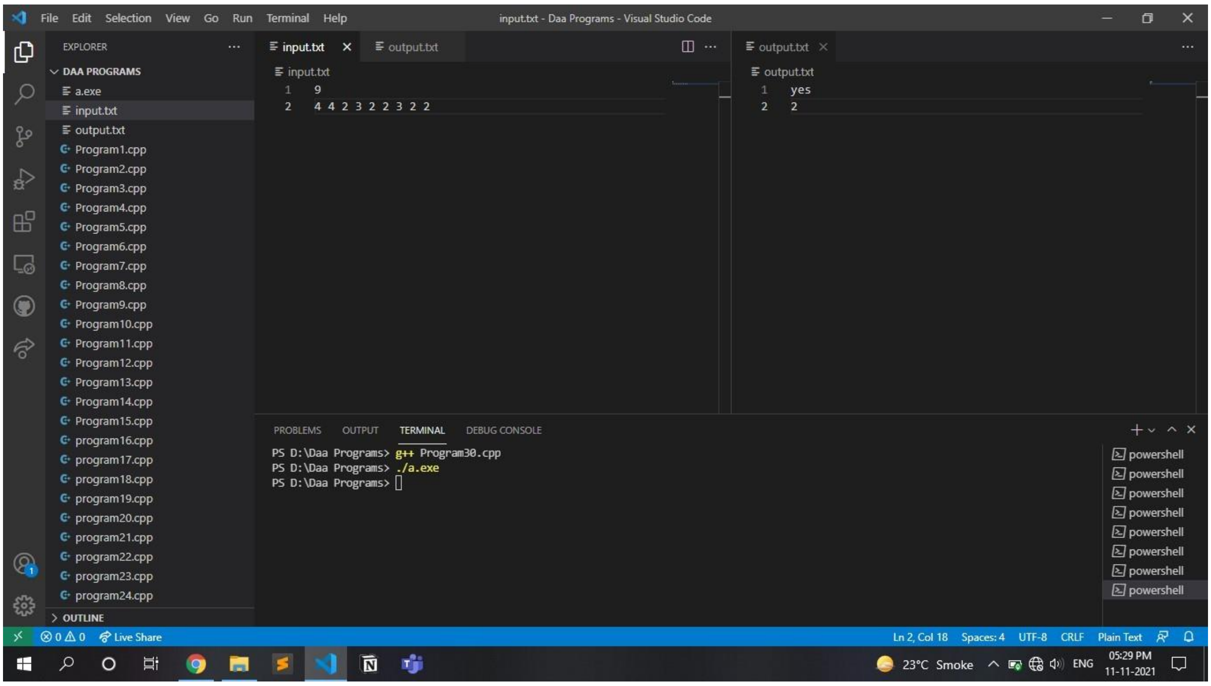
Given an unsorted array of elements, design an algorithm and implement it using a program to find whether majority element exists or not. Also find median of the array. A majority element is an element that appears more than n/2 times, where n is the size of array.

Code:

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

string majorityElement(int *arr, int n)
{
        int count = 1, max_ele = -1, temp = arr[0], ele, f=0;
        for(int i=1;i<n;i++)
        {
                if(temp==arr[i])
                {
                        count++;
                }
                else
                {
                        count = 1;
                        temp = arr[i];
                }
                if(max_ele<count)
                {
                        max_ele = count;
                        ele = arr[i];
                        if(max_ele>(n/2))
                        {
                                f = 1;
                                break;
                        }
                }
        }
```

```
    return (f==1 ? "yes" : "no");
}

int main()
{

    #ifndef ONLINE_JUDGE
        freopen("input.txt","r",stdin);
        freopen("output.txt","w",stdout);
    #endif

    int n;
    cin>>n;
    int arr[n];

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

    sort(arr, arr+n);

    cout<<majorityElement(arr, n)<<endl;

    if(n%2 == 0){
        cout<<(arr[n/2 -1]+arr[n/2])/2;
    }
    else{
        cout<<arr[n/2];
    }

    return 0;
}
```

OUTPUT



input.txt

```
9
4 4 2 3 2 2 3 2 2
```

output.txt

```
yes
2
```

```
PS D:\Daa Programs> g++ Program30.cpp
PS D:\Daa Programs> ./a.exe
PS D:\Daa Programs>
```