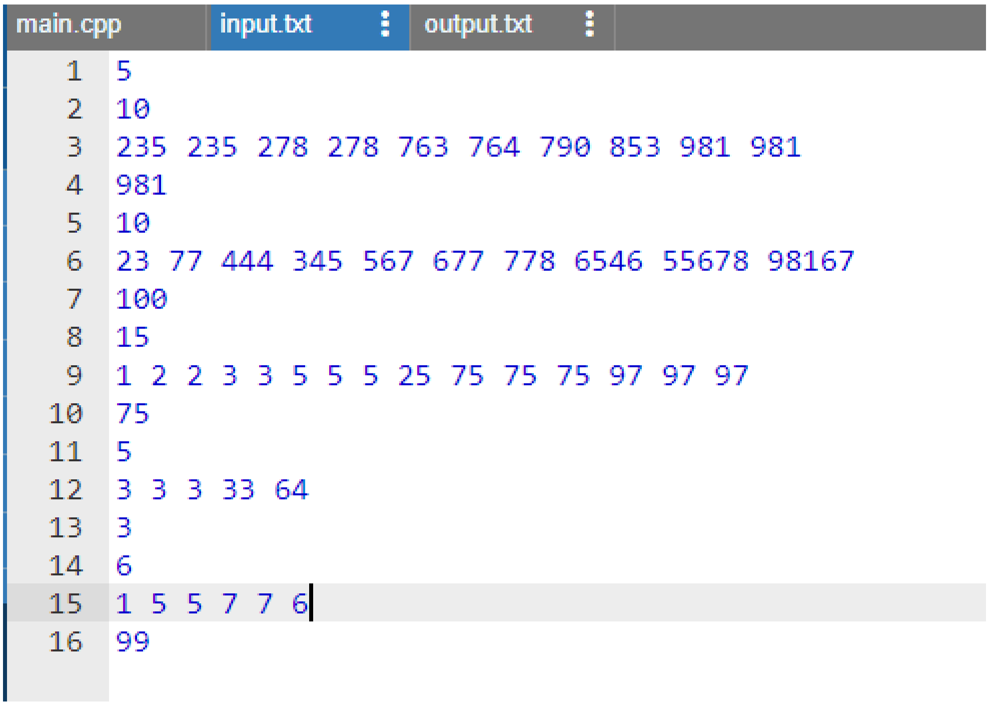
WEEK 2

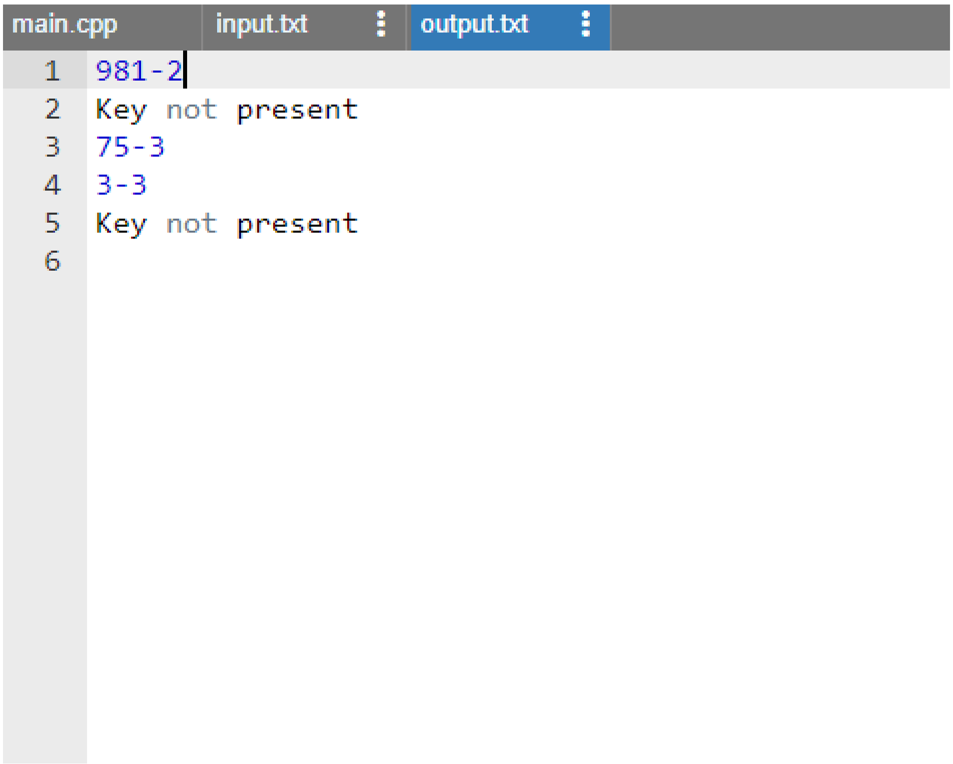
1 .Given a sorted array of positive integers containing few duplicate elements, design an algorithm and implement it using a program to find whether the given key element is present in the array or not. If present, then also find the number of copies of given key. (Time Complexity = O(log n))

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

                                                                    //function
void frequency(vector<int>& nums, int target,ofstream &fout) {
   int start=-1,end=-1;
   int l=0 , r =nums.size()-1;
                                                          // find rightmost element
   while(l<=r){
      int mid =(l+r)/2;
      if(nums[mid]<=target){
         if(nums[mid]==target){
         end =mid;
         }
         l=mid+1;
      }
      else{
         r=mid-1;
      }

   }
                                                          // element not found at all
   if(end==-1){
      fout<<"Key not present\n";
      return;
   }
```

```
                                                                  //find leftmost limit
    l=0 ,r =nums.size()-1;

    while(l<=r){
        int mid =(l+r)/2;
        if(nums[mid]>=target){
            if(nums[mid]==target){
                start =mid;
            }
            r =mid-1;
        }
        else{
            l=mid+1;
        }

    }
    fout<<target<<'-'<<end-start+1<<'\n';
}



int main() {
                                                            // Open input and output files
    ifstream fin("input.txt");
    ofstream fout("output.txt");
                                                      // Check if files are opened successfully
    if (!fin.is_open() || !fout.is_open()) {
        cout << "Error occurred while opening files.\n";
        return 0;
    }
    int t;
                                                                  // Number of test cases
    fin >> t;
    while (t--) {
        int n;
        fin >> n;
        vector<int> arr(n);

                                                                 // Input array elements
        for (int i = 0; i < n; i++) {
            fin >> arr[i];
        }

        int key;
                                                                  // Key to be searched
        fin >> key;
                                               // Perform binary search and output the result
        frequency(arr, key, fout);
    }
                                                             // Close input and output files
    fin.close();
    fout.close();
}
```



```
5
10
235 235 278 278 763 764 790 853 981 981
981
10
23 77 444 345 567 677 778 6546 55678 98167
100
15
1 2 2 3 3 5 5 5 25 75 75 75 97 97 97
75
5
3 3 3 33 64
3
6
1 5 5 7 7 6
99
```

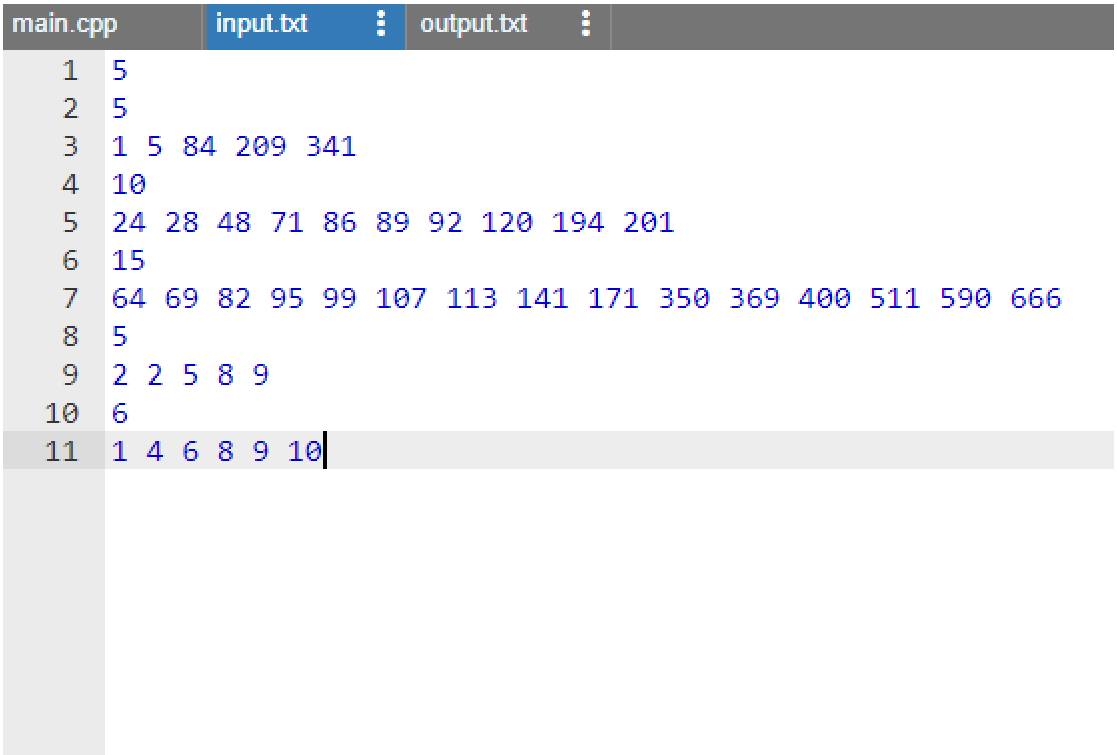


```
981-2
Key not present
75-3
3-3
Key not present
```

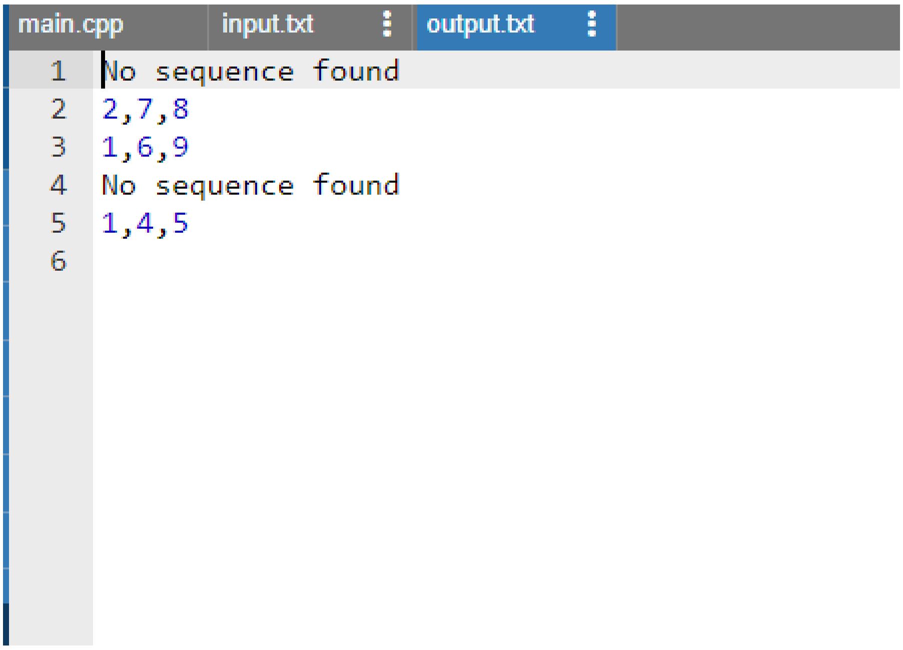
2 . Given a sorted array of positive integers, design an algorithm and implement it using a program to find three indices i, j, k such that arr[i] + arr[j] = arr[k].

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

                                                                    //Binary search
int binary_search(vector<int>&arr,int low,int high,int k)
{
   while(low<=high){
      int mid = (low+high)/2;
      if(arr[mid]==k) return mid;
      else if(arr[mid]>k) high= mid-1;
      else low =mid+1;
   }
   return -1;
}

                                                                        //function
void threeseq(vector<int>& nums,ofstream &fout) {
   int n =nums.size();
                              // select two indices and find third by using binary search.
   for(int i=0;i<n;i++){
      for(int j=i+1;j<n;j++){
         int k = binary_search(nums,j+1,n-1,nums[i]+nums[j]);
         if(k!=-1){
            fout<<i+1<<','<<j+1<<','<<k+1<<'\n';
            return;
         }
      }
   }
```

```
    fout<<"No sequence found\n";
}

int main() {
                                                                    // Open input and output files
    ifstream fin("input.txt");
    ofstream fout("output.txt");
                                                            // Check if files are opened successfully
    if (!fin.is_open() || !fout.is_open()) {
        cout << "Error occurred while opening files.\n";
        return 0;
    }
    int t;
                                                                          // Number of test cases
    fin >> t;
    while (t--) {
        int n;
        fin >> n;
        vector<int> arr(n);

                                                                          // Input array elements
        for (int i = 0; i < n; i++) {
            fin >> arr[i];
        }
                                                       // Perform binary search and output the result
        threeseq(arr, fout);
    }
                                                                   // Close input and output files
    fin.close();
    fout.close();
}
```



```
5
5
1 5 84 209 341
10
24 28 48 71 86 89 92 120 194 201
15
64 69 82 95 99 107 113 141 171 350 369 400 511 590 666
5
2 2 5 8 9
6
1 4 6 8 9 10
```

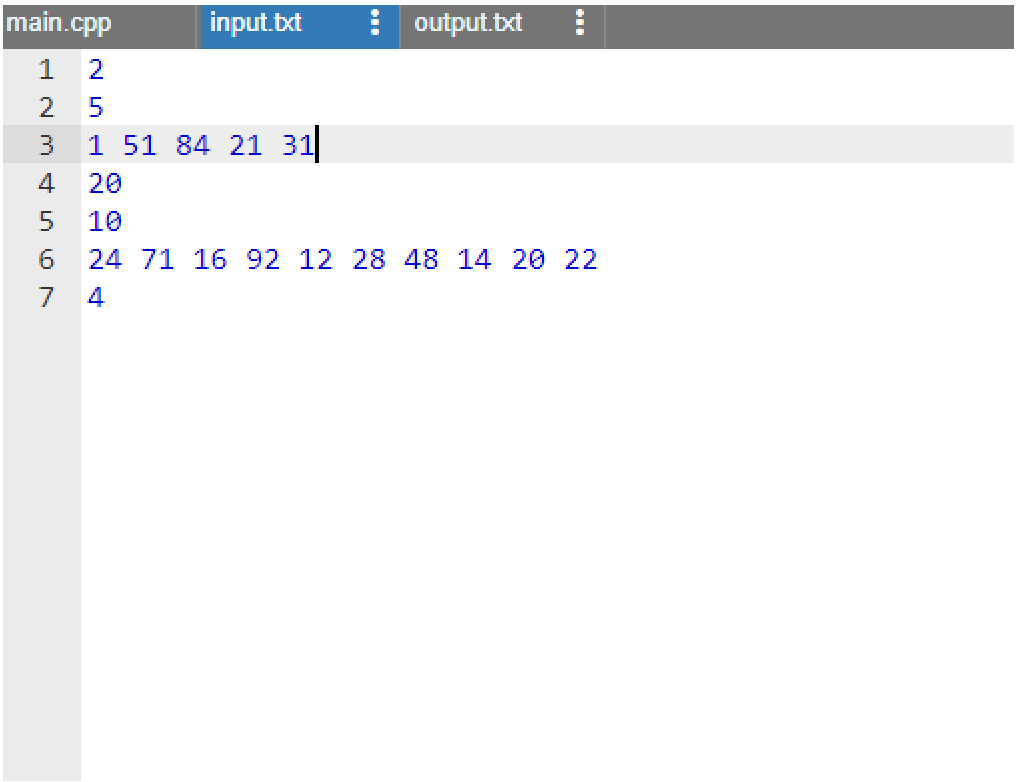


```
No sequence found
2,7,8
1,6,9
No sequence found
1,4,5

```

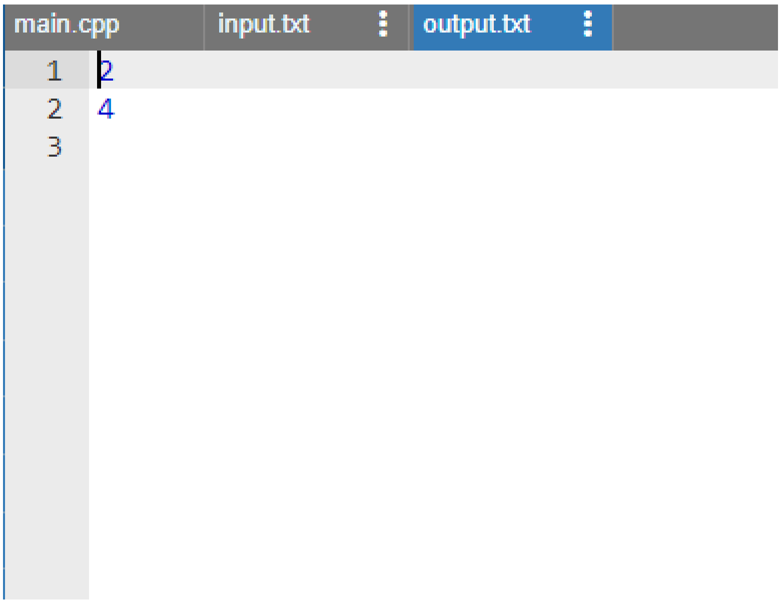
3 . Given an array of non-negative integers, design an algorithm and a program to count the number of pairs of integers such that their difference is equal to a given key, K.

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

int binary_search(vector<int>&nums,int s,int e,int target){
    int start=-1,end=-1;
                                                                    // find rightmost element
    int l =s,r=e;
    while(l<=r){
        int mid =(l+r)/2;
        if(nums[mid]<=target){
            if(nums[mid]==target){
            end =mid;
            }
            l=mid+1;
        }
        else{
            r=mid-1;
        }

    }
                                                                    // element not found at all
    if(end==-1){
        return 0;
    }
                                                                    //find leftmost limit
    l=s,r=e;
    while(l<=r){
        int mid =(l+r)/2;
        if(nums[mid]>=target){
            if(nums[mid]==target){
                start =mid;
            }
            r =mid-1;
        }
        else{
            l=mid+1;
        }

    }
    return end-start+1;
}

                                                                                    //function
void frequency(vector<int>& nums, int target,ofstream &fout) {
    int ans=0;
    int n = nums.size();
    sort(nums.begin(),nums.end());
    for(int i=0;i<n;i++){
        // a-b = k so  b = k-a
        ans+= binary_search(nums,i+1,n-1,nums[i]-target);
        // b-a =k so b =k+a
        ans+= binary_search(nums,i+1,n-1,nums[i]+target);

    }
    fout<<ans<<endl;

}
```

```
int main() {
                                                        // Open input and output files
    ifstream fin("input.txt");
    ofstream fout("output.txt");
                                               // Check if files are opened successfully
    if (!fin.is_open() || !fout.is_open()) {
        cout << "Error occurred while opening files.\n";
        return 0;
    }
    int t;
                                                               // Number of test cases
    fin >> t;
    while (t--) {

        int n;
        fin >> n;
        vector<int> arr(n);

                                                               // Input array elements
        for (int i = 0; i < n; i++) {
            fin >> arr[i];
        }

        int key;
                                                                 // Key to be searched
        fin >> key;
        // Perform binary search and output the result
        frequency(arr, key, fout);
    }
    // Close input and output files
    fin.close();
    fout.close();
}
```



```
2
5
1 51 84 21 31
20
10
24 71 16 92 12 28 48 14 20 22
4
```



```
2
4
```