ASSIGNMENT 1

AIM: TO CREATE ADT TO PERFORM THE FOLLOWING SET OPERATIONS:

1. ADD (NEW ELEMENT) PLACE A VALUE IN A SET.
2. REMOVE(ELEMENT).
3. RETURNS TRUE IF ELEMENT IS IN COLLECTION.
4. SIZE() RETURNS NUMBER OF VALUES IN A COLLECTION.
5. INTERSECTION OF TWO SETS.
6. UNION OF TWO SETS.
7. DIFFERENCE BETWEEN TWO SETS
8. SUBSET.

OBJECTIVE: TO IMPLEMENT THE " SET " CONCEPT.

THEORY : A **set** is an abstract data type that can store unique values, without any particular order. It is a computer implementation of the mathematical concept of a finite set. Unlike most other collection types, rather than retrieving a specific element from a set, one typically tests a value for membership in a set. One may define the operations of the algebra of sets:

- union(*S*,*T*) : returns the union of sets *S* and *T*.
- intersection(*S*,*T*) : returns the intersection of sets *S* and *T*.
- difference(*S*,*T*) : returns the difference of sets *S* and *T*.
- subset(*S*,*T*) : a predicate that tests whether the set *S* is a subset of set *T*.

ALGORITHM:

Union*:*

1) Initialize union U as empty.
2) Copy all elements of first array to U.
3) Do following for every element x of second array:
…..a) If x is not present in first array, then copy x to U.
4) Return U.

Intersection:

1) Initialize intersection I as empty.
2) Do following for every element x of first array
…..a) If x is present in second array, then copy x to I.
4) Return I.

CODE:

```
#include <iostream>


using namespace std;


int set1[100],set2[100];


class Set{
private:
   int arr[100];
   int currLength;
public:
   Set(){
      currLength = 0;
   }
   Set(const Set &s){
      for(int i = 0 ;i<s.currLength; i++){
         arr[i] = s.arr[i];
      }
      currLength = s.currLength;
   }
```

```
void input(){
    cout<<"Enter no. of elements to be entered : ";
    int no;
    cin>>no;
    if(no<=100){
        cout<<"Enter the numbers : ";
        for(int i =0;i<no;i++){
            cin>>arr[i];
        }
        currLength = no;
    }
}

void add(int val){
    if(currLength<=100){
        arr[currLength] = val;
    }
    currLength++;
}

void del(int val){
    bool found = false;
    for(int i = 0; i<currLength; i++){
        if(arr[i] == val){
```

```
            found = true;
            int j = i;
            for(j = i; j<currLength-1; j++){
               arr[j] = arr[j+1];
            }
            arr[j] = 0;
            currLength--;
         }
      }
      if(!found){
         cout<<"The number is not present in the set."<<endl;
      }
   }

   void findNo(int val){
      bool found = false;
      for(int i = 0; i<currLength; i++){
         if(arr[i] == val){
            cout<<val<<" found at location "<<i<<endl;
            found = true;
         }
      }
      if(!found){
         cout<<"The number is not present in the set."<<endl;
      }
```

```
    }

    bool findNoPresence(int val){
        bool found = false;
        for(int i = 0; i<currLength; i++){
            if(arr[i] == val){
                found = true;
            }
        }
        return found;
    }

    void print(){
        for(int i=0;i<currLength; i++){
            cout<<arr[i]<<"  ";
        }
        cout<<endl;
    }

    int getIndexVal(int index){
        return arr[index];
    }

    int sizeofset(){
        return currLength;
```

```
    }

};

void setsUnion(Set set1, Set set2){
    Set ans;
    for(int i = 0; i<set1.sizeofset(); i++){
        ans.add(set1.getIndexVal(i));
    }
    for(int j  = 0 ; j<set2.sizeofset(); j++){
        if(!ans.findNoPresence(set2.getIndexVal(j))){
            ans.add(set2.getIndexVal(j));
        }
    }
    cout<<"Union : ";
    ans.print();
}

void setsIntersection(Set set1, Set set2){
    Set ans;
    for(int i = 0 ; i<set1.sizeofset(); i++){
        if(set2.findNoPresence(set1.getIndexVal(i))){
            ans.add(set1.getIndexVal(i));
        }
    }
```

```
    cout<<"Intersection : ";
    ans.print();
}

void setsDifference(Set set1, Set set2){
    Set ans = set1;
    for(int i = 0; i<set2.sizeofset(); i++){
        if(ans.findNoPresence(set2.getIndexVal(i))){
            ans.del(set2.getIndexVal(i));
        }
    }
    cout<<"Difference : ";
    ans.print();
}

void subset(Set set1, Set set2){
    int matches = 0;
    if(set2.sizeofset() <= set1.sizeofset()){
        for(matches = 0; matches<set2.sizeofset();matches++){
            if(!set1.findNoPresence(set2.getIndexVal(matches))){
                break;
            }
        }
    }
    if(matches == set2.sizeofset()){
```

```
        cout<<"Set 2 is subset of Set 1."<<endl;
    }
    else{cout<<"Set 2 is not a subset of Set 1."<<endl;}



}
int main()
{
    Set set1,set2;
    char ch;
    do{
        cout<<"::::::::::::::::::::::::::::::::::::"<<endl;
        cout<<"1.Create set"<<endl<<"2.Add integer"<<endl<<"3.Delete
integer"<<endl<<"4.Find Position of integer"<<endl;

cout<<"5.Union"<<endl<<"6.Intersection"<<endl<<"7.Difference"<<endl<<"8.Subset"
<<endl<<"9.Print Set 1"<<endl<<"10.Print Set 2"<<endl;
        cout<<endl<<"Enter your choice : ";
        int choice;
        cin>>choice;
        switch(choice){
            case 1 : set1.input();
            break;
            case 2 :
                cout<<"Enter number to be inserted : ";
                int no1;
```

```
        cin>>no1;
        set1.add(no1);
    break;
    case 3 :
        cout<<"Enter number to be deleted : ";
        int no2;
        cin>>no2;
        set1.del(no2);
    break;
    case 4 :
        cout<<"Enter number : ";
        int no3;
        cin>>no3;
        set1.findNo(no3);
    break;
    case 5 :
        if(set2.sizeofset() == 0){
            set2.input();
        }
        setsUnion(set1,set2);
    break;
    case 6 :
        if(set2.sizeofset() == 0){
            set2.input();
        }
```

```
            setsIntersection(set1,set2);
        break;
        case 7 :
            if(set2.sizeofset() == 0){
                set2.input();
            }
            setsDifference(set1,set2);
        break;
        case 8 :
            if(set2.sizeofset() == 0){
                set2.input();
            }
            subset(set1,set2);
        break;
        case 9 :
            set1.print();
        break;
        case 10 :
            set2.print();
        break;
        default : cout<<"Wrong input !!"<<endl;
      }
      cout<<"Do you want to continue ? [Y/N]";
      cin>>ch;
   }while(ch=='y' || ch=='Y');
```

```
    return 0;
}
```

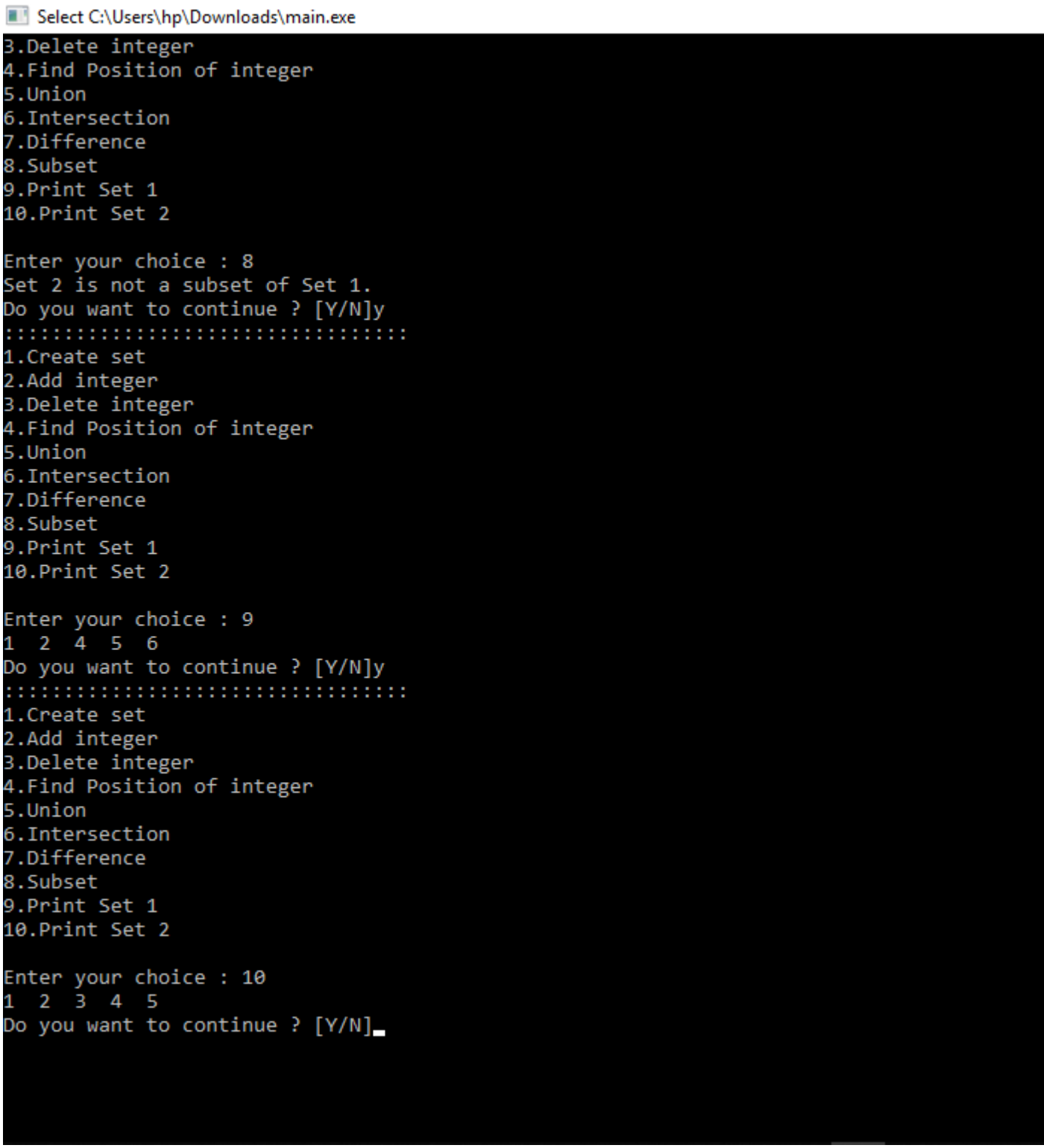
OUTPUT:

Select C:\Users\hp\Downloads\main.exe

```
::::::::::::::::::::::::::::::::
1.Create set
2.Add integer
3.Delete integer
4.Find Position of integer
5.Union
6.Intersection
7.Difference
8.Subset
9.Print Set 1
10.Print Set 2

Enter your choice : 1
Enter no. of elements to be entered : 5
Enter the numbers : 1
2
3
4
5
Do you want to continue ? [Y/N]y
::::::::::::::::::::::::::::::::
1.Create set
2.Add integer
3.Delete integer
4.Find Position of integer
5.Union
6.Intersection
7.Difference
8.Subset
9.Print Set 1
10.Print Set 2

Enter your choice : 2
Enter number to be inserted : 6
Do you want to continue ? [Y/N]y
::::::::::::::::::::::::::::::::
1.Create set
2.Add integer
3.Delete integer
4.Find Position of integer
5.Union
6.Intersection
7.Difference
8.Subset
9.Print Set 1
10.Print Set 2
```



CONCLUSION: We saw all the algorithms the STL offers to operate on sets, that are collections of sorted elements, in the general sense.