**ASSIGNMENT 1**

**AIM**:To create ADT that implement the "set" concept. a. Add (newElement) -Place a value into the set b. Remove (element) c. Contains (element) Return true if element is in collection d. Size () Return number of values in collection e. Intersection of two sets f. Union of two sets g. Difference between two sets h.Subset.

**OBJECTIVE**: TO IMPLEMENT THE “ SET ” CONCEPT.

**THEORY** : A **set** is an [abstract data type](https://en.wikipedia.org/wiki/Abstract_data_type" \o "Abstract data type) that can store unique values, without any particular [order](https://en.wikipedia.org/wiki/Sequence" \o "Sequence). It is a computer implementation of the [mathematical](https://en.wikipedia.org/wiki/Mathematics" \o "Mathematics) concept of a [finite set](https://en.wikipedia.org/wiki/Finite_set" \o "Finite set). Unlike most other [collection](https://en.wikipedia.org/wiki/Collection_(abstract_data_type)" \o "Collection (abstract data type)) 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](https://en.wikipedia.org/wiki/Algebra_of_sets" \o "Algebra of sets):

* union(*S*,*T*): returns the [union](https://en.wikipedia.org/wiki/Union_(set_theory)" \o "Union (set theory)) of sets *S* and *T*.
* intersection(*S*,*T*): returns the [intersection](https://en.wikipedia.org/wiki/Intersection_(set_theory)" \o "Intersection (set theory)) of sets *S* and *T*.
* difference(*S*,*T*): returns the [difference](https://en.wikipedia.org/wiki/Difference_(set_theory)" \o "Difference (set theory)) of sets *S* and *T*.
* subset(*S*,*T*): a predicate that tests whether the set *S* is a [subset](https://en.wikipedia.org/wiki/Subset" \o "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 s1[50],s2[50],n1,n2;

void insert()

{

int ch,val;

cout<<"Enter set no 1 or 2 - \n";

cin>>ch;

if(ch==1)

{

cout<<"Enter ur element\n";

cin>>val;

s1[n1]=val;

n1++;

}

else if(ch==2)

{

cout<<"Enter ur element\n";

cin>>val;

s1[n2]=val;

n2++;

}

}

void display()

{

cout<<"Elements of set1\n"<<endl;

for(int i=0;i<n1;i++)

cout<<s1[i]<<" ";

cout<<"Elements of set2\n"<<endl;

for(int i=0;i<n2;i++)

cout<<s2[i]<<" ";

}

void remove()

{

int ch,key,i,count=0;

cout<<"Enter set no 1 or 2 - \n";

cin>>ch;

if(ch==1)

{

cout<<"Enter element\n";

cin>>key;

for (i=0;i<n1;i++)

if (s1[i]==key)

break;

if(i<n1)

{

n1=n1-1;

for(int j=i;j<n1;j++)

s1[j]=s1[j+1];

}

}

else if(ch==2)

{

cout<<"Enter element\n";

cin>>key;

for (i=0;i<n2;i++)

if (s2[i]==key)

break;

if(i<n2)

{

n2=n2-1;

for(int j=i;j<n2;j++)

s2[j]=s2[j+1];

}

}

}

void contains()

{

int ch,key,f=0;

cout<<"Enter set no 1 or 2 - \n";

cin>>ch;

cout<<"Enter the element to be found\n";

cin>>key;

if(ch==1)

{

for(int i=0;i<n1;i++)

{

if(s1[i]==key)

{

f=1;

}

}

if(f==0)

cout<<"Element not found\n Element is "<<key<<"\n";

if (f==1)

cout<<"Element found\n Element is "<<key<<"\n";

}

else if(ch==2)

{

for(int i=0;i<n2;i++)

{

if(s2[i]==key)

{

f=1;

}

}

if(f==0)

cout<<"Element not found\n Element is "<<key<<"\n";

if (f==1)

cout<<"Element found\n Element is "<<key<<"\n";

}

}

void size()

{

int ch,cnt=0;

cout<<"Enter set no 1 or 2 -\n ";

cin>>ch;

if(ch==1)

{

for(int i=0;i<n1;i++)

cnt++;

cout<<"Size of set 1 is "<<cnt;

}

else if(ch==2)

{

for(int i=0;i<n2;i++)

cnt++;

cout<<"Size of set 2 is "<<cnt;

}

}

void intersection()

{

int i,j,s3[50];

for(int i=0;i<n1;i++)

{

for(int j=0;j<n2;j++)

{

if(s1[i]==s2[j])

{

s3[i]=s1[i];

cout<<s3[i]<<" ";

}

}

}

}

void uni()

{

int i,j,s3[50];

for(int i=0;i<n1;i++)

{

for(int j=0;j<n2;j++)

{

if(s1[i]!=s2[j])

{

s3[i]=s2[j];

}

}

cout<<s1[i]<<" ";

}

}

int main()

{

cout<<"Enter size of set 1 and set 2 - ";

cin>>n1;

cin>>n2;

cout<<"Enter elements of set1"<<endl;

for(int i=0;i<n1;i++)

cin>>s1[i];

cout<<"Enter elements of set2"<<endl;

for(int i=0;i<n2;i++)

cin>>s2[i];

int ch;

while(1)

{

cout<<"\n1.Insert\n2.Remove\n3.Contains\n4.Size\n5.Display\n6.Intersection\n7.Union\n";

cout<<"Enter your choice - \n";

cin>>ch;

switch(ch)

{

case 1:

insert();

break;

case 2:

remove();

break;

case 3:

contains();

break;

case 4:

size();

break;

case 5:

display();

break;

case 6:

intersection();

break;

case 7:

uni();

break;

default:

cout<<"Enter correct choice";

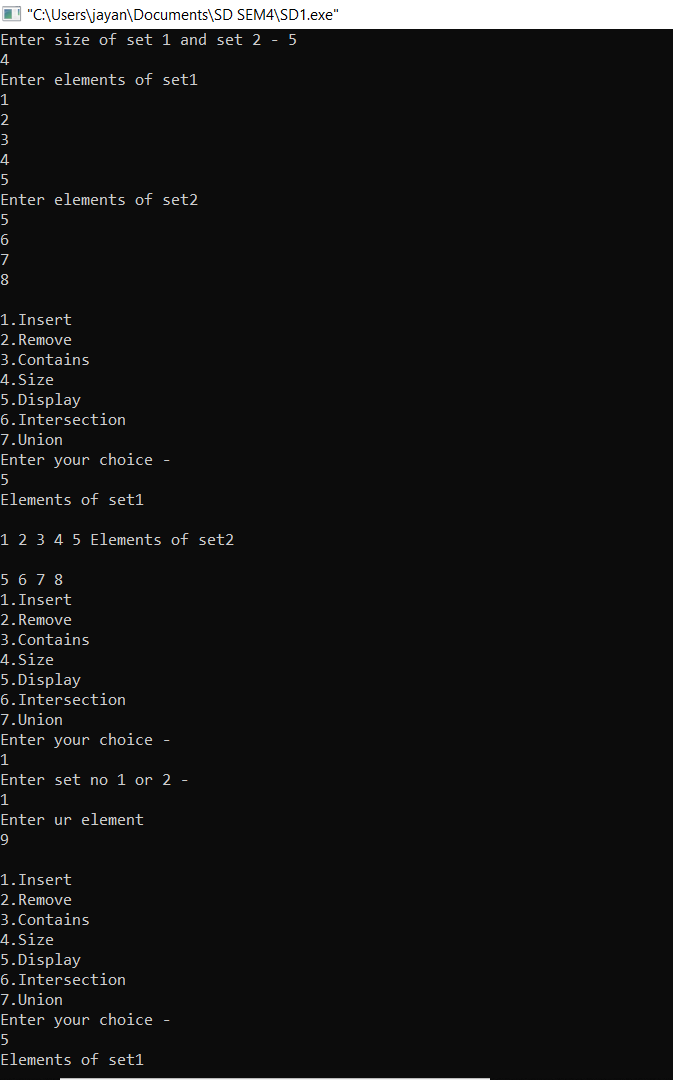
}

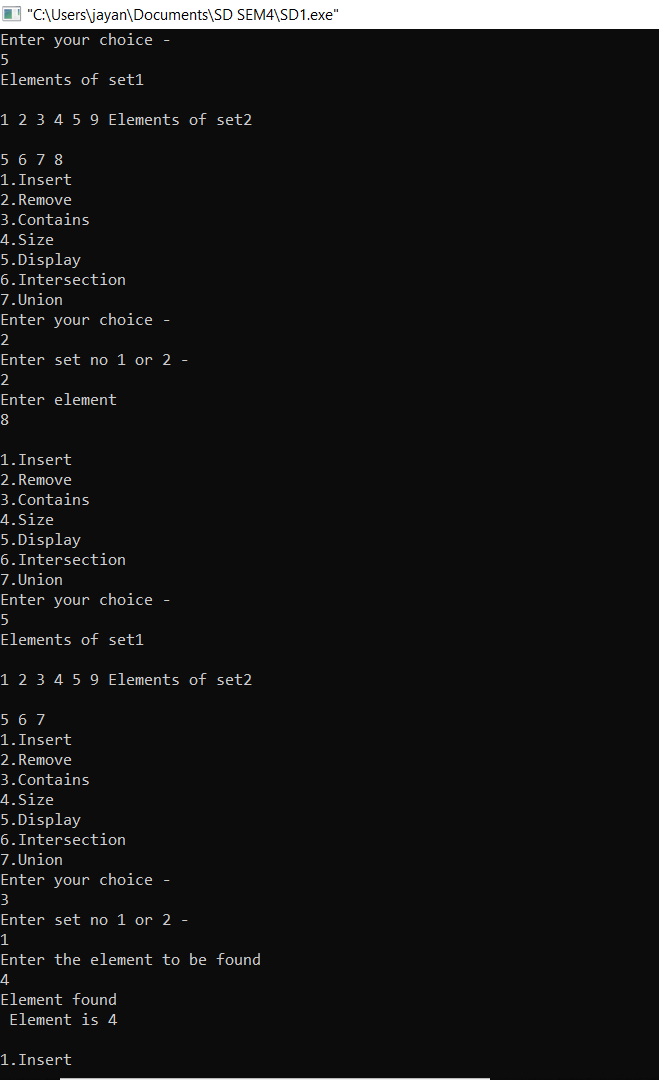
}

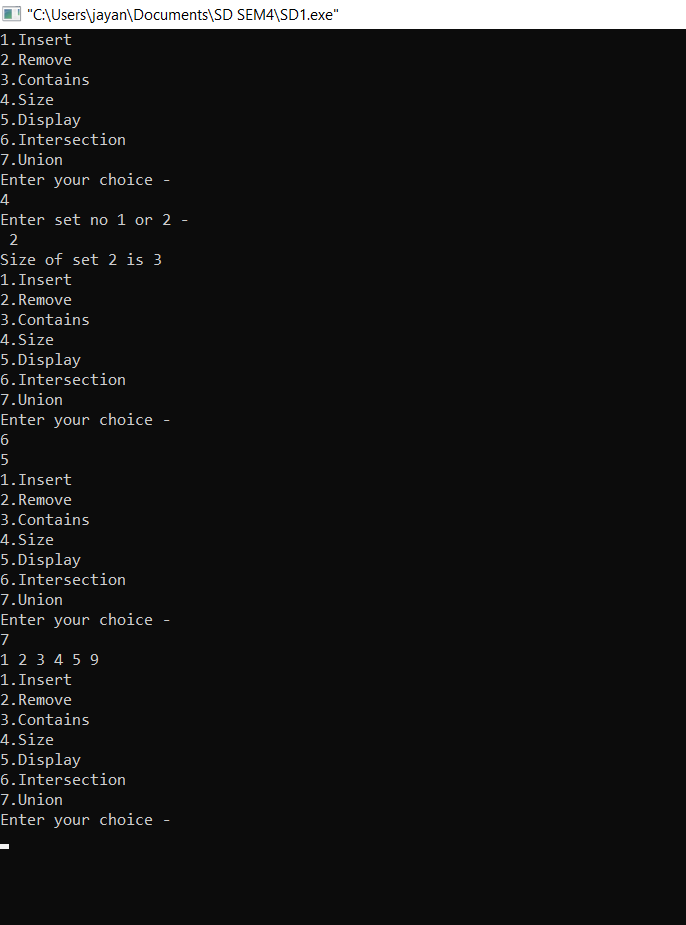
return 0;

}

**OUTPUT:**

****

****

****

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