ASSIGNMENT 1

Aim: To create ADT that implements 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 theory.

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.

Program:

```
#include<iostream>
#define max 30
using namespace std;
void create(int set1[],int n)
{
set1[0]=0;
for(int i=1;i<=n;i++)
{
cin>>set1[i];
}
set1[0]=n;
}
void display(int set1[])
{
int n=set1[0];
for(int i=1;i<=n;i++)
{
cout<<set1[i]<<"\t";
} cout<<endl;
}
int member(int set1[],int x)
{
int n=set1[0];
for(int i=1;i<=n;i++)
{
```

```
if(set1[i]==x)
return 1;
}
return 0;
}
void intersection(int set1[],int set2[],int set3[])
{
set3[0]=0;
int n=set1[0];
for(int i=1;i<=n;i++)
{
if(member(set2,set1[i]))
{
set3[0]++;
set3[set3[0]]=set1[i];
}
}
}
void unions(int set1[],int set2[],int set3[])
{
int n=set1[0];
set3[0]=n;
for(int i=1;i<=n;i++)
{
set3[i]=set1[i];
}
```

```
n=set2[0];
for(int i=1;i<=n;i++)
{
if(!member(set3,set2[i]))
{
set3[0]++;
set3[set3[0]]=set2[i];
}
}
}
void diff(int set1[],int set2[],int set3[])
{
set3[0]=0;
int n=set1[0];
for(int i=1;i<=n;i++)
{
if(!member(set2,set1[i]))
{
set3[0]++;
set3[set3[0]]=set1[i];
}
}
}
int subset(int set1[],int sset[])
{
int n=sset[0];
```

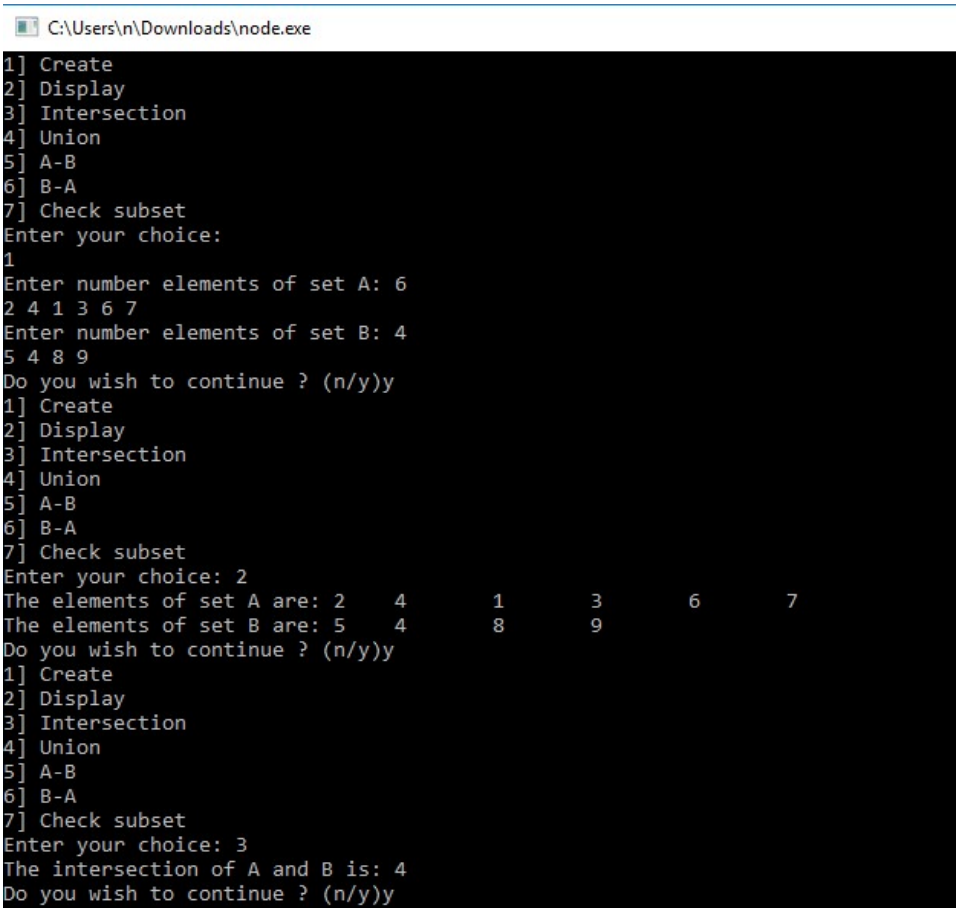
```
int flag=0;
for(int i=1;i<=n;i++)
{
if(!member(set1,sset[i]))
{
flag=1;
break;
}
}
if(flag==1)
return 0;
else
return 1;
}

int main()
{
int set1[max],set2[max],set_u[max],set_int[max],set_diff[max],set_s[max];
char c;
int choice;
do
{

cout<<"1] Create\n2] Display\n3] Intersection\n4] Union\n5] A-B\n6] B-A\n7] Check
subset\n";
cout<<"Enter your choice: ";
cin>>choice;
```

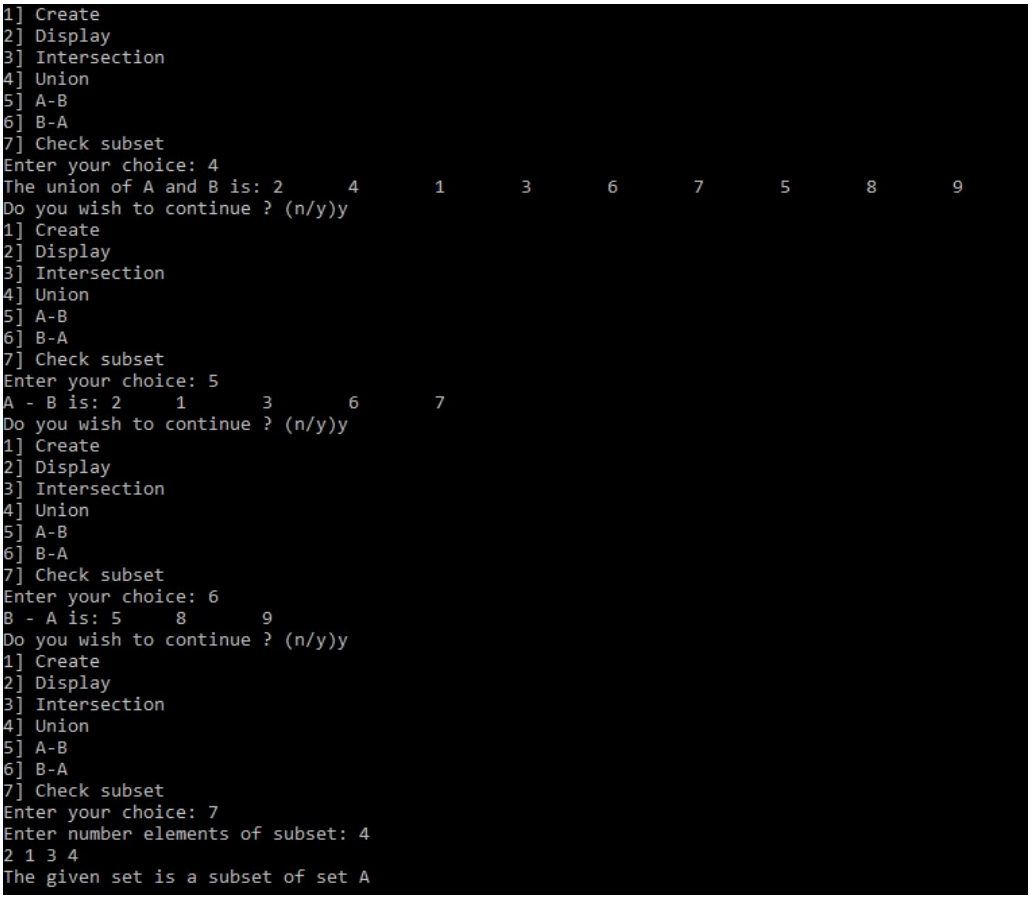
```
switch(choice)
{
case 1:{
int s1,s2;
cout<<"Enter number elements of set A: ";
cin>>s1;
create(set1,s1);
cout<<"Enter number elements of set B: ";
cin>>s2;
create(set2,s2);
}
break;
case 2:{
cout<<"The elements of set A are: ";
display(set1);
cout<<"The elements of set B are: ";
display(set2);
}
break;
case 3:{
intersection(set1,set2,set_int);
cout<<"The intersection of A and B is: ";
display(set_int);
}
break;
case 4:{
```

```
unions(set1,set2,set_u);
cout<<"The union of A and B is: ";
display(set_u);
}
break;
case 5:{
diff(set1,set2,set_diff);
cout<<"A - B is: ";
display(set_diff);
}
break;
case 6:{
diff(set2,set1,set_diff);
cout<<"B - A is: ";
display(set_diff);
}
break;
case 7:{
int n;
cout<<"Enter number elements of subset: ";
cin>>n;
create(set_s,n);
if(subset(set1,set_s) && subset(set2,set_s))
cout<<"The given set is a subset of both sets\n";
else if(subset(set2,set_s))
cout<<"The given set is a subset of set B\n";
```

```
else if(subset(set1,set_s))

cout<<"The given set is a subset of set A\n";

else

cout<<"The given set is not a subset of any set\n";

}

break;}

cout<<"Do you wish to continue ? (n/y)";

cin>>c;

}while(c!='y'|| 'Y');

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.