# ASSIGNMENT NO.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 study the different set operations. To get the thorough understanding of the concepts of sets and the various operations performed on it.

## Theory:-

**Set** is a container implemented in C++ language in STL and has a concept similar to how set is defined in mathematics. The facts that separates set from the other containers is that is it contains only the **distinct elements**and elements can be traversed in sorted order. Having the strong hold on sets is useful in competitive programming and solving algorithmic problems. The insertion and deletion in STL sets are discussed in this article.Sets have the most impact in mathematical set theory. These theories are used in many kinds of proofs, structures, and abstract algebra. Creating relations from different sets and codomains are also an important applications of sets.

In computer science, set theory is useful if you need to collect data and do not care about their multiplcity or their order. In databases, especially for relational databases, sets are very useful. There are many commands that finds unions, intersections, and differences of different tables and sets of data.

EQUAL SETS-

Two sets are said to be equal if both have same elements. For example A = {1, 3, 9, 7} and B = {3, 1, 7, 9} are equal sets. SUBSET:-• A set A is said to be subset of another set B if and only if every element of set A is also a part of other set B. Denoted by '⊆'. 'A ⊆ B ' denotes A is a subset of B.

SIZE OF A SET:-

Size of a set can be finite or infinite. Size of the set S is known as Cardinality number, denoted as |S|. Example: Let A be a set of odd positive integers less than 10. Solution : A = {1,3,5,7,9}, Cardinality of the set is 5, i.e.,|A| = 5.

UNION :•

Union of the sets A and B, denoted by A ∪ B, is the set of distinct element belongs to set A or set B, or both.

INTERSECTION:• The intersection of the sets A and B, denoted by A ∩ B, is the set of elements belongs to both A and B i.e. set of the common element in A and B SET

DIFFERENCE:-•

Difference between sets is denoted by 'A – B', is the set containing elements of set A but not in B. i.e all elements of A except the element of B.

## ***ALGORITHM:***

FOR INTERSECTION:•

Step 1: Take an empty set (intersection set) Step 2: pass each element of set-2 and the entire set-1 to the function member() Step 3:if it returns true, Add that element to the intersection set FOR

UNION:•

Step 1: Take an empty set (union set)

Step 2: copy all the elements of set1 to this new set

Step 3: traverse through the set2 and pass each element of set-2 along with the entire set-1 to the function member(), and if it returns false then add that specified element to the union set

FOR CONTAINS:•

Step1 : take the number as input which you want to search

Step 2 : enter 1 for searching in set-1 or 2 for searching in set-2

Step 3 : initialise i=0

Step 4: traverse the set-1 or set-2 till the end depending on whether the input was 1 or 2 after passing the element and that set to the function member()

Step 5: if element found then display element is present C Department of Computer

4 FOR SUBSET:•

Step 1: enter 1 if you want to check if set 2 is subset of 1, or enter 2 if you want to check if set-1 is subset of set-2

Step 2: depending on input we will traverse the set(which has to be the subset) until its end, by passing each element of this set and other set to the function member()

Step 3: if member() returns true, then continue else return false
Step 4: if false, then display “it is a subset” else display “it is not”

FOR DIFFERENCE :•
Step 1: Initialise the difference set to 0, difference set contains all the element which are in set-1 but not in set-2
Step 2: traverse the entire set-1 and, pass each element of this set and the the set-2 to the function member()
Step 3: if it returns false then add this element to the difference set

FOR REMOVE:•
Step 1: enter 1 or 2 for removing element from set-1 or set-2 respectively
Step 2: enter the index from which you want to remove the element
Step 3: if , entered position is less than the size of the set then move all the elements to their left from the position at which you want to remove the element and just decrease the size of the set else, entered position is equal to the size of the set then just decrease the size

FOR SIZE:•
step 1: show the 0th index of the set which contains the size of our set

## Program Code:-

```
#include <iostream>

using namespace std;


const int MAX=50;

template<class T>

class SET

{

        T data[MAX];
```

```
        int n;
public:
        SET()
{
                n=-1;
}
        bool insert(T);
        bool remove(T);
        bool contains(T);
        int size();
        void print();
        void input(int num);
        SET unionS(SET,SET);
        SET intersection(SET,SET);
        SET difference(SET,SET);
};

template<class T>
void SET<T>::input(int num)
{
        T element;
        for(int i=0;i<num;i++)
        {
                cout<<"\nEnter Element: "<<i+1;
```

```
                cin>>element;
                insert(element);
        }
}
template<class T>
void SET<T>::print()
{
        for(int i=0;i<=n;i++)
                cout<<" "<<data[i];
}
template<class T>
SET<T> SET<T>::unionS(SET<T> s1,SET<T> s2)
{
        SET<T> s3;

        int flag=0;
        int i=0;
        for(i=0;i<=s1.n;i++)
        {
                s3.insert(s1.data[i]);
        }
        for(int j=0;j<=s2.n;j++)
        {
                flag=0;
```

```
                for(i=0;i<=s1.n;i++)
                {
                        if(s1.data[i]==s2.data[j])
                        {
                                flag=1;
                                break;
                        }
                }
                if(flag==0)
                {
                        s3.insert(s2.data[j]);

                }
        }

        return s3;
}

template<class T>
SET<T> SET<T>::difference(SET<T> s1,SET<T> s2)
{
        SET<T> s3;
        int flag=1;
        for(int i=0;i<=s1.n;i++)
```

```
        {
                for(int j=0;j<=s2.n;j++)
                {
                        if(s1.data[i]==s2.data[j])
                        {
                                flag=0;
                                break;
                        }
                        else flag=1;
                }
                if(flag==1)
                {
                        s3.insert(s1.data[i]);

                }
        }
        return s3;
}

template<class T>
SET<T> SET<T>::intersection(SET<T> s1,SET<T> s2)
{
        SET<T> s3;
        for(int i=0;i<=s1.n;i++)
```

```
        {
                for(int j=0;j<=s2.n;j++)
                {
                        if(s1.data[i]==s2.data[j])
                        {
                                s3.insert(s1.data[i]);
                                break;
                        }
                }
        }
        return s3;
}
template<class T>
bool SET<T>::insert(T element)
{
        if(n>=MAX)
        {
                cout<<"\nOverflow.SET is full.\n";
                return false;
        }
        data[++n]=element;
        return true;
}
```

```
template<class T>
bool SET<T>::remove(T element)
{
	if(n==-1)
	{
		cout<<"Underflow. Cannot perform delete operation on empty SET.";
		return false;
	}
	for(int i=0;i<=n;i++)
	{
		if(data[i]==element)
		{
			for(int j=i;i<=n;j++)
			{
				data[j]=data[j+1];
			}
			return true;
		}
	}
	//data[n--]=0;
	return false;
}
template<class T>
bool SET<T>::contains(T element)
```

```
{
        for(int i=0;i<=n;i++)
        {
                if(data[i]==element)
                        return true;
        }
        return false;
}
template<class T>
int SET<T>::size()
{
        return n+1;
}
int main() {

        SET<int> s1,s2,s3;
        int choice;
        int element;
        cout<<"\nEnter number of elements in SET1:";
        cin>>element;//element is used for taking size
        s1.input(element);
        cout<<"\nEnter number of elements in SET2:";
        cin>>element;//element is used for taking size
        s2.input(element);
```

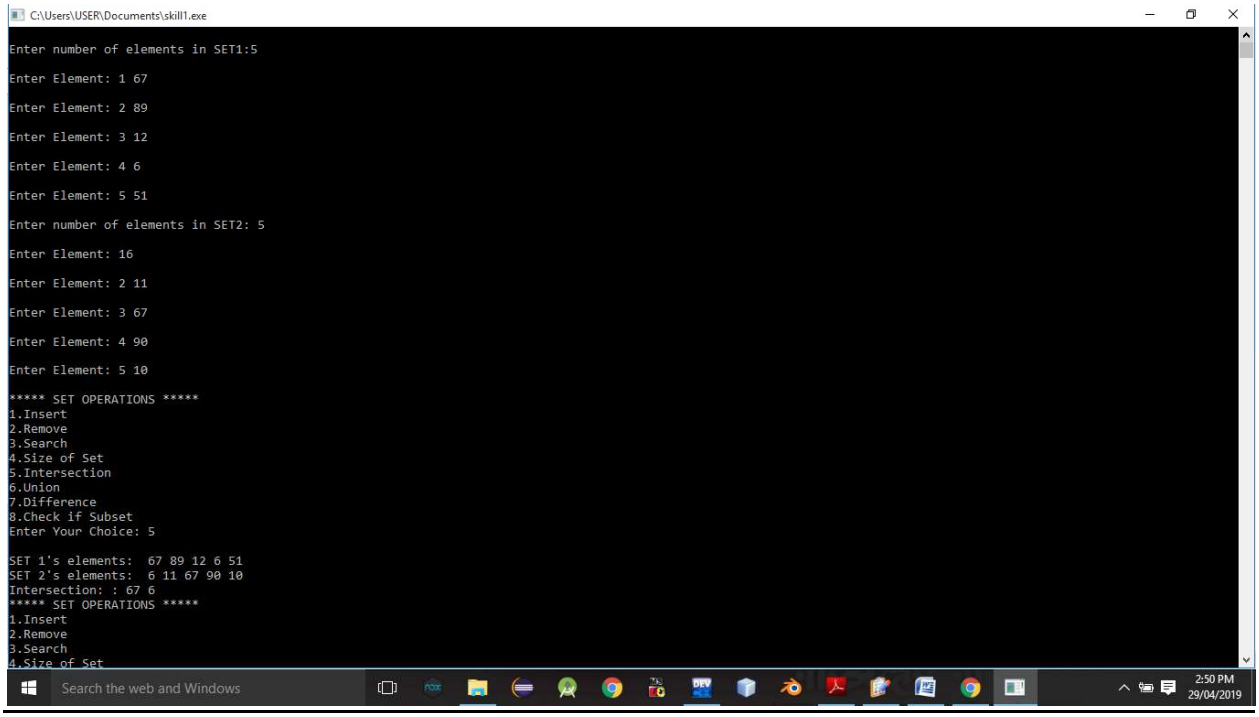
```
do
{
        cout<<"\n***** SET OPERATIONS *****"
                <<"\n1.Insert"
                <<"\n2.Remove"
                <<"\n3.Search"
                <<"\n4.Size of Set"
                <<"\n5.Intersection"
                <<"\n6.Union"
                <<"\n7.Difference"
                <<"\n8.Check if Subset"
                <<"\nEnter Your Choice: ";
        cin>>choice;
        switch(choice)
        {
        case 1:
                cout<<"\nEnter Element: ";
                cin>>element;
                if(s1.insert(element))
                {
                        cout<<element<<" inserted";
                }
                else
                {
```

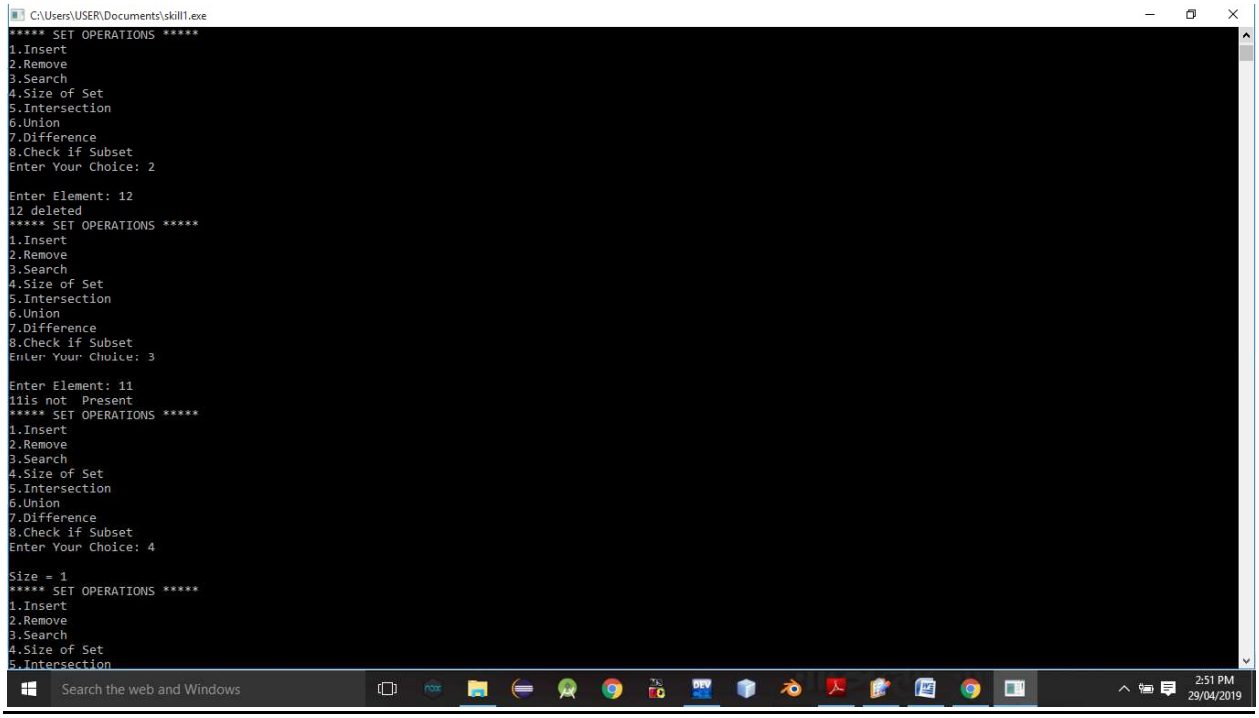
```
                cout<<"Insertion Failed";
            }
            break;
        case 2:
            cout<<"\nEnter Element: ";
            cin>>element;
            if(s1.remove(element))
            {
                cout<<element<<" deleted";
            }
            else
            {
                cout<<"Deletion Failed";
            }
            break;
        case 3:
            cout<<"\nEnter Element: ";
            cin>>element;
            if(s1.contains(element))
            {
                cout<<element<<" is present";
            }
            else
            {
```

```
                cout<<element<<"is not  Present";
        }
        break;
case 4:
        cout<<"\nSize = "<<s1.size();
        break;
case 5:
        s3=s1.intersection(s1,s2);
        cout<<"\nSET 1's elements: ";
        s1.print();
        cout<<"\nSET 2's elements: ";
        s2.print();
        cout<<"\nIntersection: :";
        s3.print();
        break;

case 6:

        s3=s1.unionS(s1,s2);
        cout<<"\nSET 1's elements: ";
        s1.print();
        cout<<"\nSET 2's elements: ";
        s2.print();
        cout<<"\nUnion :";
```

```
                    s3.print();
                    break;
                case 7:
                    s3=s1.difference(s1,s2);
                    cout<<"\nSET 1's elements: ";
                    s1.print();
                    cout<<"\nSET 2's elements: ";
                    s2.print();
                    cout<<"\nDifference :";
                    s3.print();
                    break;

                }
        }while(choice!=0);
        return 0;
}
```

## Output Screenshots:-



## Conclusion:-

Thus,we have studied different operations on set ADT.