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.

Theory:-

Sets are containers that store unique elements following a specific order.

In a set, the value of an element also identifies it (the value is itself the *key*, of type T), and each value must be unique. The value of the elements in a set cannot be modified once in the container (the elements are always const), but they can be inserted or removed from the container. Internally, the elements in a set are always sorted following a specific *strict weak ordering* criterion indicated by its internal comparison object (of type Compare).

set containers are generally slower than unordered_set containers to access individual elements by their *key*, but they allow the direct iteration on subsets based on their order.

Sets are typically implemented as *binary search trees*.

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;
}
```

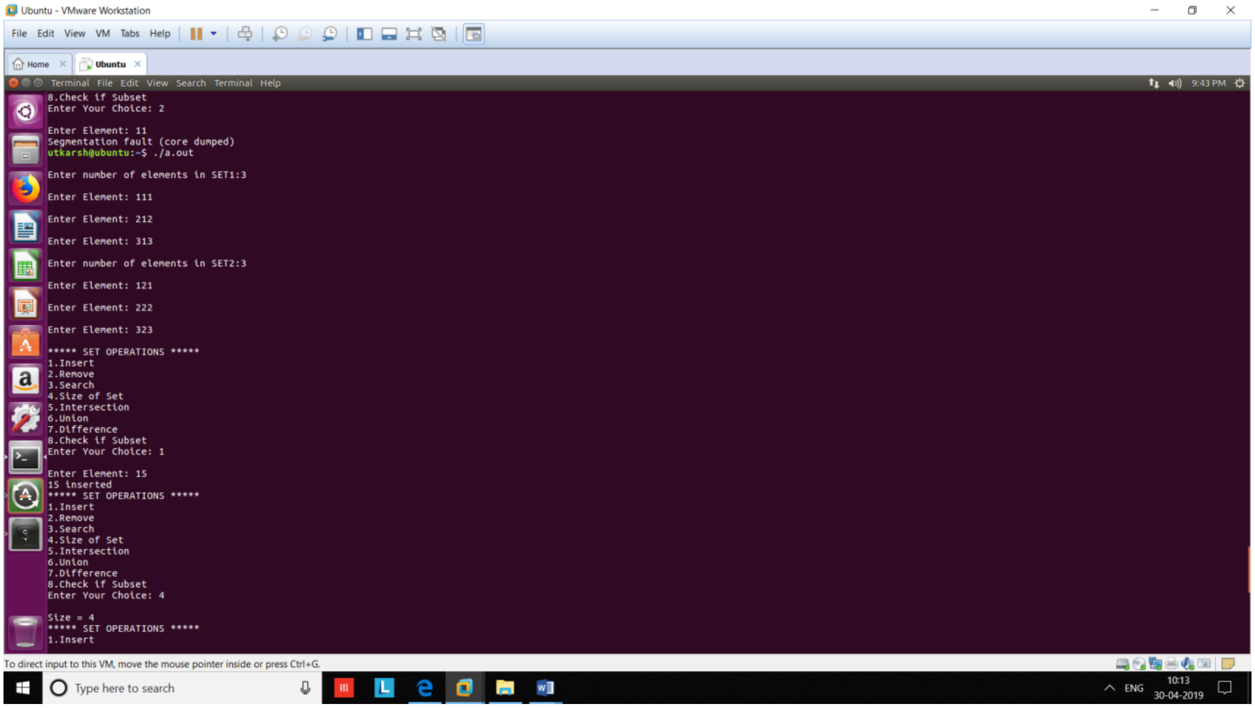
```
template<class T>
SET<T> SET<T>::difference(SET<T> s1,SET<T> s2)
{
        SET<T> s3;
        int flag=1;
        for(int i=0;i<=s1.n;i++)
        {
                for(int j=0;j<=s2.n;j++)
                {
                        if(s1.data[i]==s2.data[j])
                        {
           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

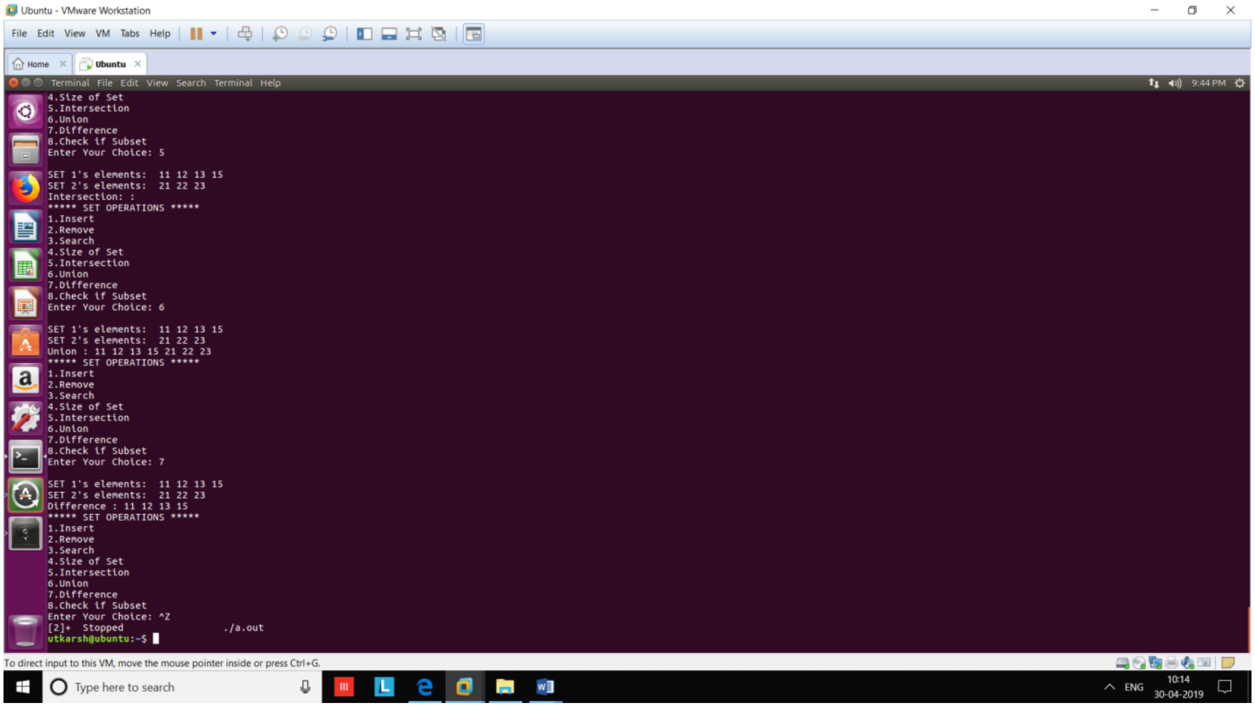
```
                        s3.print();

                        break;


                }

        }while(choice!=0);

        return 0;

}
```

Output Screenshots:-



Ubuntu - VMware Workstation
File Edit View VM Tabs Help
Home
Ubuntu
Terminal File Edit View Search Terminal Help
9:43 PM
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Type here to search
ENG
10:13
30-04-2019

```
8.Check if Subset
Enter Your Choice: 2

Enter Element: 11
Segmentation fault (core dumped)
utkarsh@ubuntu:~$ ./a.out

Enter number of elements in SET1:3

Enter Element: 111

Enter Element: 212

Enter Element: 313

Enter number of elements in SET2:3

Enter Element: 121

Enter Element: 222

Enter Element: 323

***** SET OPERATIONS *****
1.Insert
2.Remove
3.Search
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check if Subset
Enter Your Choice: 1

Enter Element: 15
15 inserted
***** SET OPERATIONS *****
1.Insert
2.Remove
3.Search
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check if Subset
Enter Your Choice: 4

Size = 4
***** SET OPERATIONS *****
1.Insert
```



Ubuntu - VMware Workstation
File Edit View VM Tabs Help
Home
Ubuntu
Terminal File Edit View Search Terminal Help
9:44 PM
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Type here to search
ENG
10:14
30-04-2019

```
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check if Subset
Enter Your Choice: 5

SET 1's elements:  11 12 13 15
SET 2's elements:  21 22 23
Intersection: :
***** SET OPERATIONS *****
1.Insert
2.Remove
3.Search
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check if Subset
Enter Your Choice: 6

SET 1's elements:  11 12 13 15
SET 2's elements:  21 22 23
Union : 11 12 13 15 21 22 23
***** SET OPERATIONS *****
1.Insert
2.Remove
3.Search
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check if Subset
Enter Your Choice: 7

SET 1's elements:  11 12 13 15
SET 2's elements:  21 22 23
Difference : 11 12 13 15
***** SET OPERATIONS *****
1.Insert
2.Remove
3.Search
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check if Subset
Enter Your Choice: ^Z
[2]+  Stopped                 ./a.out
utkarsh@ubuntu:~$ 
```

Conclusion:- Thus,we have studied different operations on set ADT.