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>
#include<bits/stdc++.h>
using namespace std;
```

```
class Set
{
   int elements;
public:
   int m[50];
   Set()
   {
      elements = 0;
      for(int i=0;i<50;i++)
      {
         m[i] = -1;
      }
   }
   void insert(int data)
   {
      m[elements] = data;
      elements++;
   }
   void remove(int data)
   {
      int flag = 0,j;
      for(int i=0;i<elements;i++)
      {
         if(m[i] == data)
         {
            flag = 1;
            for(j=i;j<elements;j++)
            {
               m[j] = m[j+1];
            }
            m[j] = -1;
            elements--;
            break;
         }
      }
      if(flag == 1)
         cout<<"Element deleted successfully "<<endl;
      else
         cout<<"Element not found "<<endl;
   }
   bool contains(int data)
   {
      int flag = 0;
```

```
    for(int i=0;i<elements;i++)
    {
      if(m[i] == data)
      {
        flag = 1;
        break;
      }
    }
    if(flag == 1)
      return true;
    else
      return false;
  }
  void setSize(int size)
  {
    elements = size;
  }
  int getSize()
  {
    return elements;
  }
  void display()
  {
    for(int i=0;i<elements;i++)
    {
      cout<<m[i]<<" ";
    }
    cout<<endl;
  }
  void sort()
  {
    int temp;
    for(int i=0;i<elements;i++)
    {
      for(int j=0;j<elements;j++)
      {
        if(m[i]<m[j])
        {
          temp = m[i];
          m[i] = m[j];
          m[j] = temp;
        }
      }
```

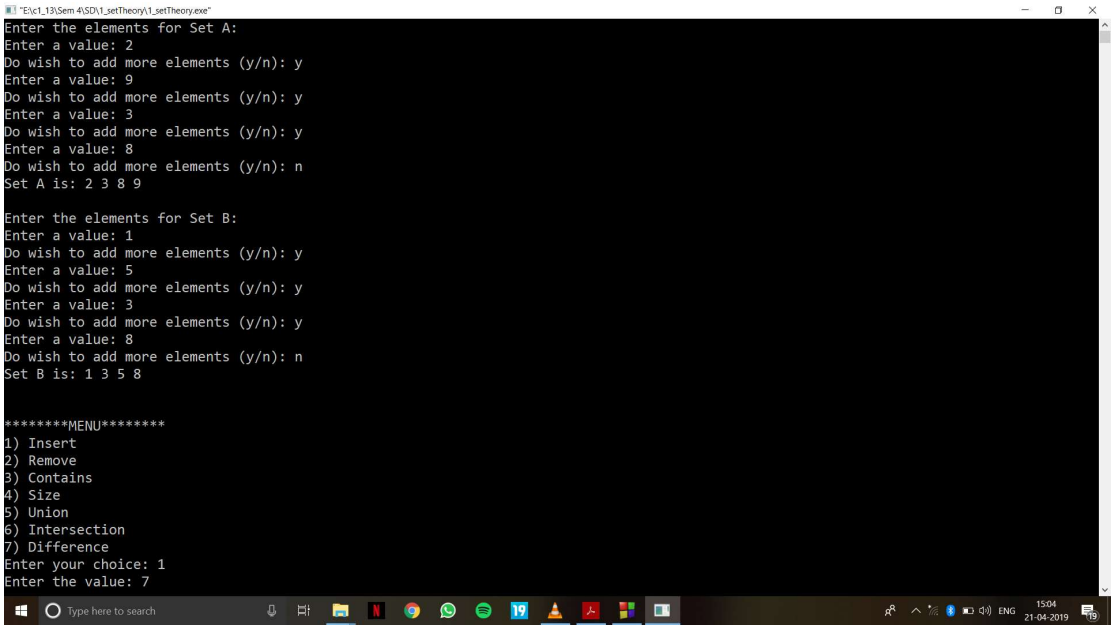
```
      }
   }
};

Set Union(Set a,Set b)
{
   Set c;
   for(int i=0;i<a.getSize();i++)
   {
      c.insert(a.m[i]);
   }
   for(int j=0;j<b.getSize();j++)
   {
      int flag = 0;
      for(int i=0;i<a.getSize();i++)
      {
         if(b.m[j] == a.m[i])
         {
            flag = 1;
            continue;
         }
      }
      if(flag == 0)
      {
         c.insert(b.m[j]);
      }
   }
   c.sort();
   return c;
}

Set Intersection(Set a,Set b)
{
   Set c;
   for(int i=0;i<a.getSize();i++)
   {
      for(int j=0;j<b.getSize();j++)
      {
         if(a.m[i] == b.m[j])
         {
            c.insert(a.m[i]);
            continue;
         }
```

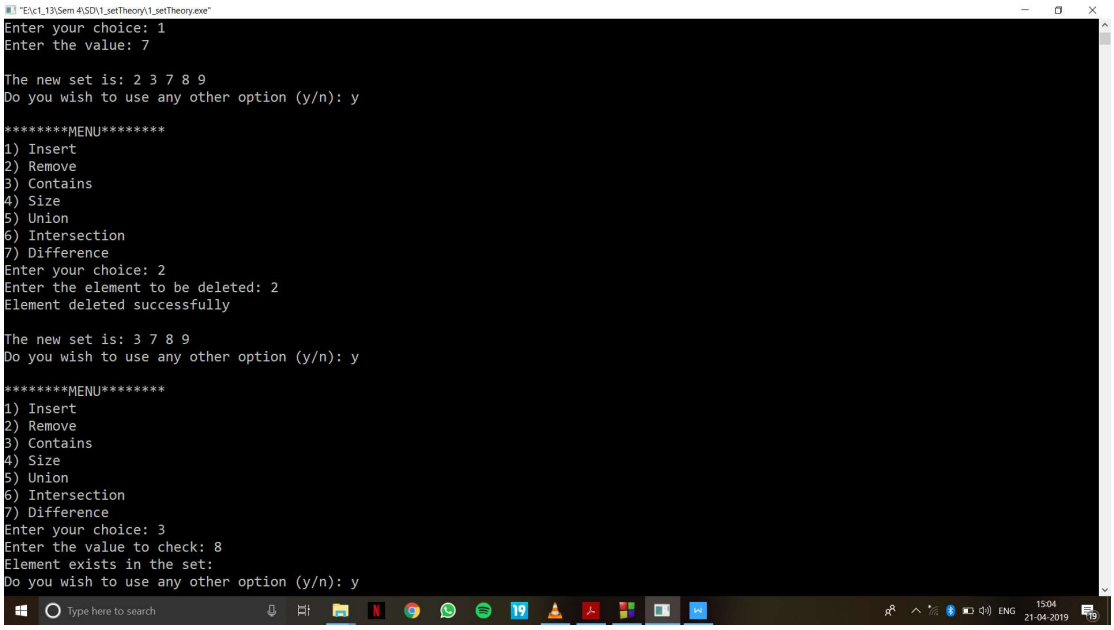
```
      }
   }
   c.sort();
   return c;
}
Set Difference(Set a,Set b)
{
   Set c;
   for(int i=0;i<a.getSize();i++)
   {
      int flag = 0;
      for(int j=0;j<b.getSize();j++)
      {
         if(a.m[i] == b.m[j])
         {
            flag = 1;
            continue;
         }
      }
      if(flag == 0)
         c.insert(a.m[i]);
   }
   c.sort();
   return c;

}
int main()
{
   Set a,b,c;
   int val,choice;
   char ch = 'y';
   cout<<"Enter the elements for Set A: "<<endl;
   while(ch == 'y')
   {
      cout<<"Enter a value: ";
      cin>>val;
      a.insert(val);
      cout<<"Do wish to add more elements (y/n): ";
      cin>>ch;
   }
   cout<<"Set A is: "; a.sort(); a.display(); cout<<endl; ch = 'y';
   cout<<"Enter the elements for Set B: "<<endl;
   while(ch == 'y')
```

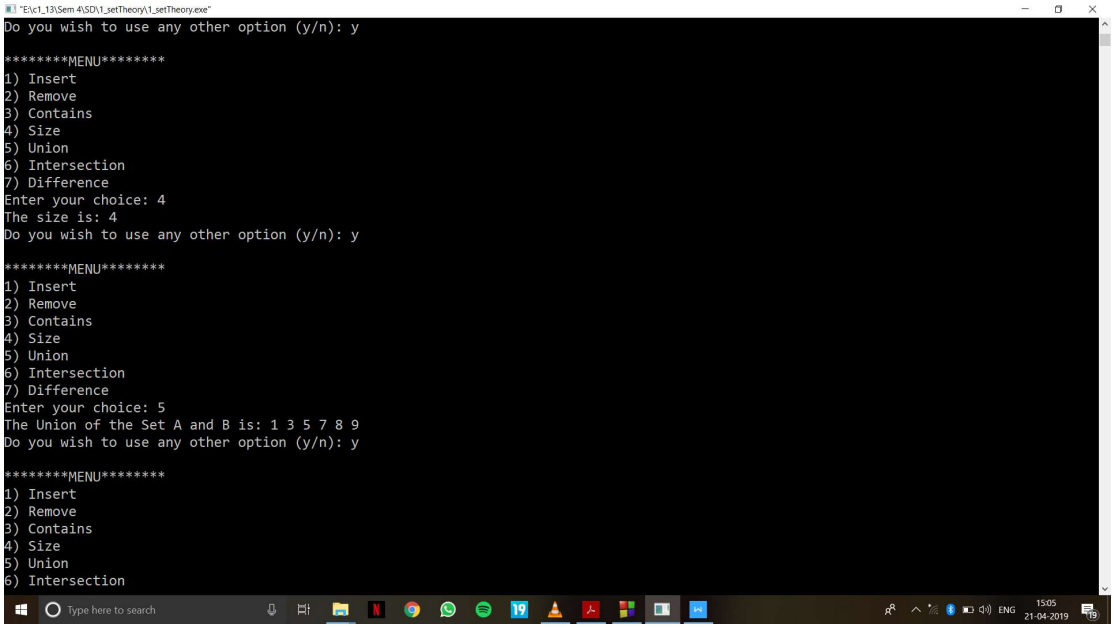
```
{
   cout<<"Enter a value: ";
   cin>>val;
   b.insert(val);
   cout<<"Do wish to add more elements (y/n): ";
   cin>>ch;
}
cout<<"Set B is: "; b.sort(); b.display(); cout<<endl; ch = 'y';
while(ch =='y')
{
   cout<<"\n********MENU********"<<endl;
   cout<<"1) Insert "<<endl;
   cout<<"2) Remove "<<endl;
   cout<<"3) Contains "<<endl;
   cout<<"4) Size "<<endl;
   cout<<"5) Union "<<endl;
   cout<<"6) Intersection "<<endl;
   cout<<"7) Difference "<<endl;
   cout<<"Enter your choice: ";
   cin>>choice;
   switch(choice)
   {
      case 1:
         cout<<"Enter the value: ";
         cin>>val;
         a.insert(val); cout<<endl; a.sort();
         cout<<"The new set is: ";
         a.display();
         break;
      case 2:
         cout<<"Enter the element to be deleted: ";
         cin>>val;
         a.remove(val); cout<<endl;
         cout<<"The new set is: ";
         a.display();

         break;
      case 3:
         cout<<"Enter the value to check: ";
         cin>>val;
         if(a.contains(val))
            cout<<"Element exists in the set: "<<endl;
         else
```

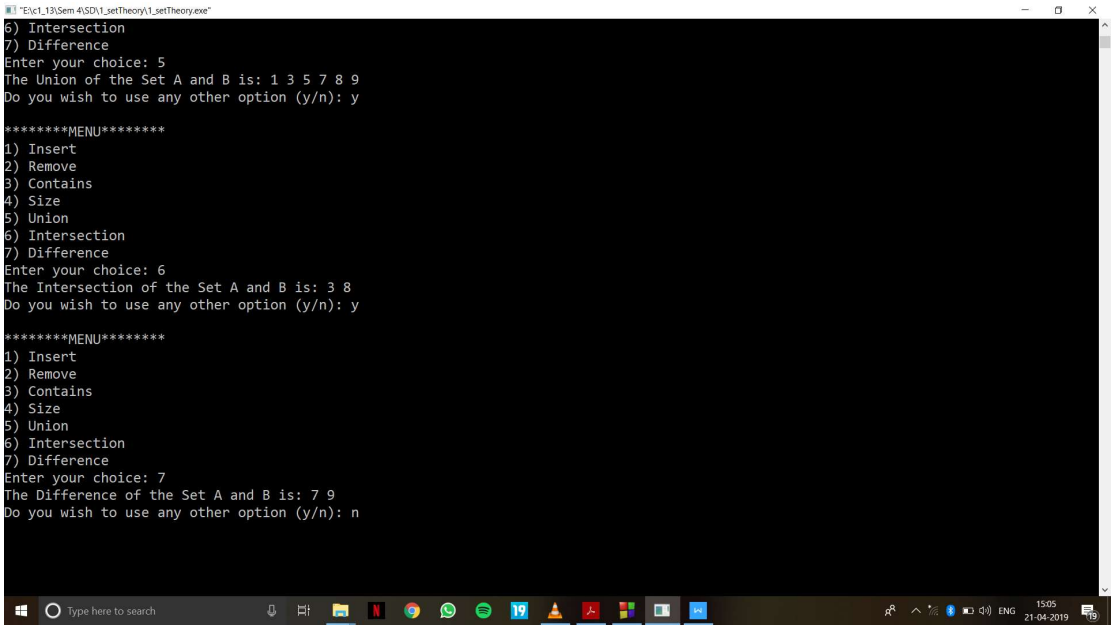
```
            cout<<"Element does not exist in the set: "<<endl;
          break;
        case 4: cout<<"The size is: "<<a.getSize()<<endl;
          break;
        case 5: cout<<"The Union of the Set A and B is: ";
          c = Union(a,b);
          c.display();
          break;
        case 6: cout<<"The Intersection of the Set A and B is: ";
          c = Intersection(a,b);
          c.display();
          break;
        case 7: cout<<"The Difference of the Set A and B is: ";
          c = Difference(a,b);
          c.display();
          break;
        default: cout<<"Wrong Choice: "<<endl;
      }
      cout<<"Do you wish to use any other option (y/n): ";
      cin>>ch;
   }
   return 0;
}
```

OUTPUT:




"E:\c1_13\Sem 4\SD\1_setTheory\1_setTheory.exe"
Enter your choice: 1
Enter the value: 7
The new set is: 2 3 7 8 9
Do you wish to use any other option (y/n): y
********MENU********
1) Insert
2) Remove
3) Contains
4) Size
5) Union
6) Intersection
7) Difference
Enter your choice: 2
Enter the element to be deleted: 2
Element deleted successfully
The new set is: 3 7 8 9
Do you wish to use any other option (y/n): y
********MENU********
1) Insert
2) Remove
3) Contains
4) Size
5) Union
6) Intersection
7) Difference
Enter your choice: 3
Enter the value to check: 8
Element exists in the set:
Do you wish to use any other option (y/n): y
Type here to search
ENG
15:04
21-04-2019


"E:\c1_13\Sem 4\SD\1_setTheory\1_setTheory.exe"
Do you wish to use any other option (y/n): y
********MENU********
1) Insert
2) Remove
3) Contains
4) Size
5) Union
6) Intersection
7) Difference
Enter your choice: 4
The size is: 4
Do you wish to use any other option (y/n): y
********MENU********
1) Insert
2) Remove
3) Contains
4) Size
5) Union
6) Intersection
7) Difference
Enter your choice: 5
The Union of the Set A and B is: 1 3 5 7 8 9
Do you wish to use any other option (y/n): y
********MENU********
1) Insert
2) Remove
3) Contains
4) Size
5) Union
6) Intersection
Type here to search
ENG
15:05
21-04-2019



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