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:-

Creating 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.

Theory:-

We shall consider ADT's that incorporate a variety of set operations. Some collections of these operations have been given special names and have special implementations of high efficiency. Some of the more common set operations are the following.

1. -3. The three procedures UNION(*A*, *B*, *C*), INTERSECTION(*A*, *B*, *C*), and DIFFERENCE(*A*, *B*, *C*) take set-valued arguments *A* and *B*, and assign the result, *A* �*B*, *A* �*B*, or *A* - *B*, respectively, to the set variable *C*.
1. We shall sometimes use an operation called *merge*, or *disjoint set union*, that is no different from union, but that assumes its operands are *disjoint* (have no members in common). The procedure MERGE(*A*, *B*, *C*) assigns to the set variable *C* the value *A* �*B*, but is not defined if *A* �*B* �Ø, i.e., if *A* and *B* are not disjoint.
2.
   The function MEMBER(*x*, *A*) takes set *A* and object *x*, whose type is the type of elements of *A*, and returns a boolean value -- true if *x* �*A* and false if *x* �*A*.
3.
   The procedure MAKENULL(*A*) makes the null set be the value for set variable *A.*
4.
   The procedure INSERT(*x*, *A*), where *A* is a set-valued variable, and *x* is an element of the type of *A*'s members, makes *x* a member of *A*. That is, the new value of *A* is *A* �{*x*}. Note that if *x* is already a member of *A*, then INSERT(*x*, *A*) does not change *A*.

5. DELETE(*x*, *A*) removes *x* from *A*, i.e., *A* is replaced by *A* - {*x*}. If *x* is not in *A* originally, DELETE(*x*, *A*) does not change *A*.
6. ASSIGN(*A*, *B*) sets the value of set variable *A* to be equal to the value of set variable *B*.
7. The function MIN(*A*) returns the least element in set *A*. This operation may be applied only when the members of the parameter set are linearly ordered. For example, MIN({2, 3, 1}) = 1 and MIN({'*a*','*b*','*c*'}) = '*a*'. We also use a function MAX with the obvious meaning.
8. EQUAL(*A*, *B*) is a function whose value is true if and only if sets *A* and *B* consist of the same elements.
9. The function FIND(*x*) operates in an environment where there is a collection of disjoint sets. FIND(*x*) returns the name of the (unique) set of which *x* is a member.

Algorithm:-

1.Insertion:

Let LA be a Linear Array (unordered) with N elements and K is a positive integer such that K<=N. Following is the algorithm where ITEM is inserted into the Kth position of LA −

1. Start

2. Set J = N

3. Set N = N+1

4. Repeat steps 5 and 6 while J >= K

5. Set LA[J+1] = LA[J]

```
6. Set J = J-1
7. Set LA[K] = ITEM
8. Stop
```

2.Deletion:-

Consider LA is a linear array with N elements and K is a positive integer such that K<=N. Following is the algorithm to delete an element available at the Kth position of LA.

```
1. Start
2. Set J = K
3. Repeat steps 4 and 5 while J < N
4. Set LA[J] = LA[J + 1]
5. Set J = J+1
6. Set N = N-1
7. Stop
```

3.Search:-

Consider LA is a linear array with N elements and K is a positive integer such that K<=N. Following is the algorithm to find an element with a value of ITEM using sequential search.

```
1. Start
2. Set J = 0
3. Repeat steps 4 and 5 while J < N
4. IF LA[J] is equal ITEM THEN GOTO STEP 6
```

```
5. Set J = J +1
6. PRINT J, ITEM
7. Stop.
```

Code:-

```
#include<iostream>
using namespace std;
void create(int set[],int n);
void display(int set[]);
void intersection(int set1[],int set2[],int set3[]);
void unions(int set1[],int set2[],int set3[]);
void diff(int set1[],int set2[],int set3[]);
int member(int set[],int x);
int subset(int set[],int sset[]);
int member(int set[],int n,int x);
#define max 30
int main()
{
        int set1[max],set2[max],set_union[max],set_int[max],set_diff[max],set_s[max];
        int c;
        do
        {
                cout<<"--------------------------------MENU---------------------------------"<<endl;
```

```
                    cout<<"1] Create\n2] Display\n3] Intersection\n4] Union\n5] A-B\n6] B-
A\n7] Check subset\n8] Exit\n";

          cout<<"_______________________________________________________________
______________"<<endl;
                    cout<<"Enter your choice: ";
                    cin>>c;
                    switch(c)
                    {
                              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_union);
                        cout<<"The union of A and B is: ";
                        display(set_union);
                    }
                    break;
            case 5:{
                        diff(set1,set2,set_diff);
                        cout<<"The difference between A and B is: ";
                        display(set_diff);
                    }
                    break;
            case 6:{
                        diff(set2,set1,set_diff);
                        cout<<"The difference between B and 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;
        }
    }while(c!=8);
```

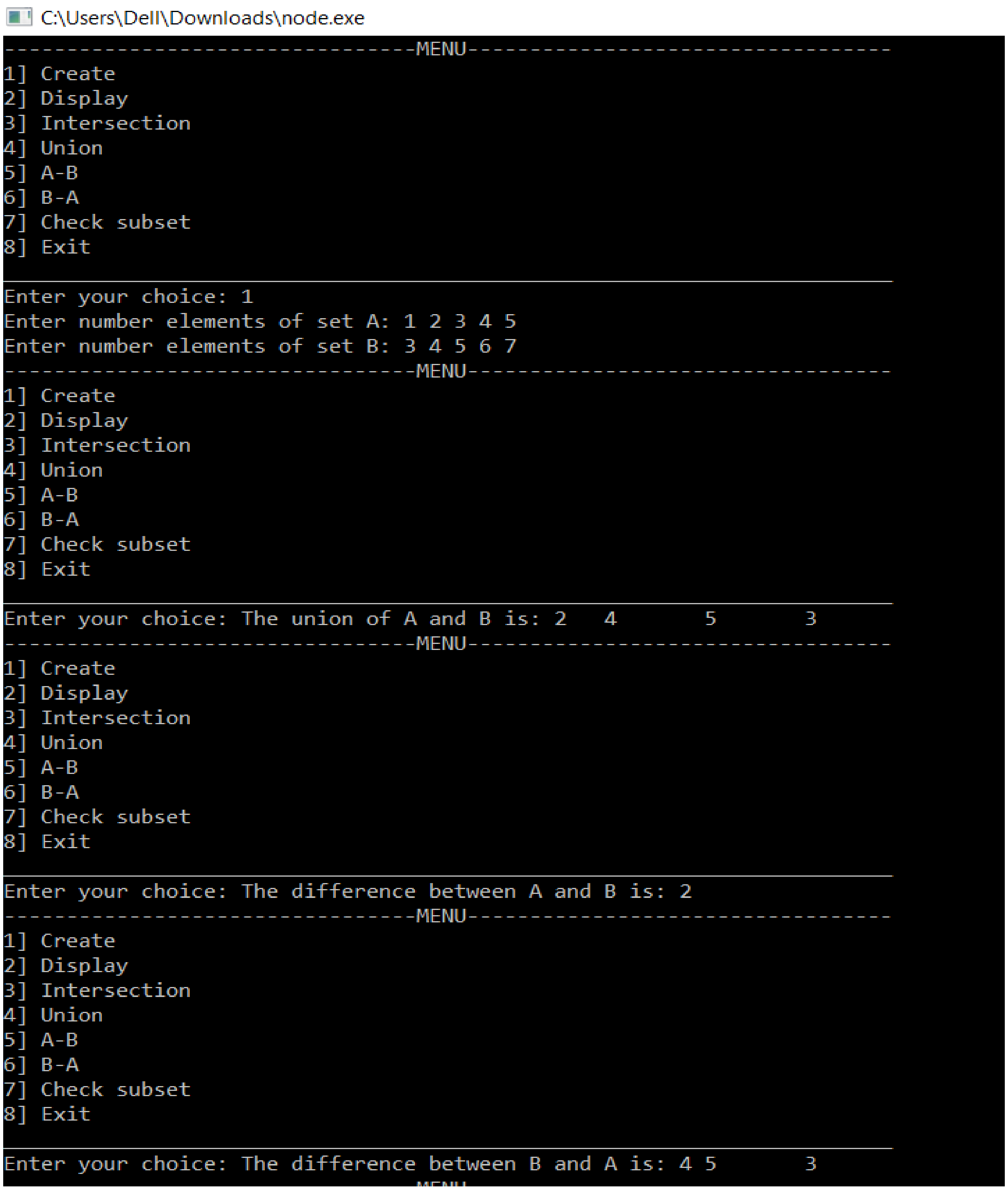
```
	return 0;
}
void create(int set[],int n)
{
	set[0]=0;
	for(int i=1;i<=n;i++)
	{
		cin>>set[i];
	}
	set[0]=n;
}
void display(int set[])
{
	int n=set[0];
	for(int i=1;i<=n;i++)
	{
		cout<<set[i]<<"\t";
	}
	cout<<endl;
}
int member(int set[],int x)
{
	int n=set[0];
```

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

Output Screenshot:-



C:\Users\Dell\Downloads\node.exe

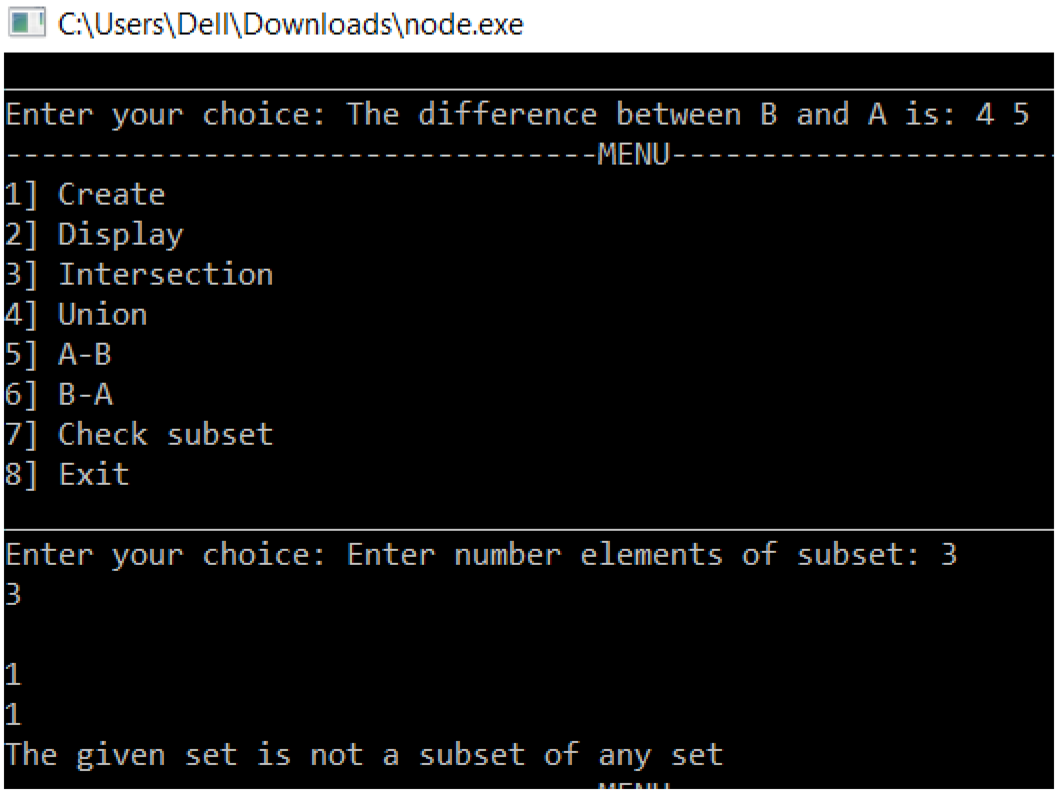
```
--------------------------------MENU----------------------------------
1] Create
2] Display
3] Intersection
4] Union
5] A-B
6] B-A
7] Check subset
8] Exit

Enter your choice: 1
Enter number elements of set A: 1 2 3 4 5
Enter number elements of set B: 3 4 5 6 7
--------------------------------MENU----------------------------------
1] Create
2] Display
3] Intersection
4] Union
5] A-B
6] B-A
7] Check subset
8] Exit

Enter your choice: The union of A and B is: 2   4       5       3
--------------------------------MENU----------------------------------
1] Create
2] Display
3] Intersection
4] Union
5] A-B
6] B-A
7] Check subset
8] Exit

Enter your choice: The difference between A and B is: 2
--------------------------------MENU----------------------------------
1] Create
2] Display
3] Intersection
4] Union
5] A-B
6] B-A
7] Check subset
8] Exit

Enter your choice: The difference between B and A is: 4 5       3
```



```
C:\Users\Dell\Downloads\node.exe

Enter your choice: The difference between B and A is: 4 5
---------------------------------MENU---------------------
1] Create
2] Display
3] Intersection
4] Union
5] A-B
6] B-A
7] Check subset
8] Exit

Enter your choice: Enter number elements of subset: 3
3

1
1
The given set is not a subset of any set
```

Conclusion:-

We Performed Various Operations of Sets by using datstructures and various alglorithms in programming language.