

ASSIGNMENT 1

1. **Aim** –To create ADT that implements the SET concept.

- a. Add (newElement) -Place a value into the set
- b. Remove (element) Remove the value
- c. Contains (element) Return true if element is in collection
- d. Size () Return number of values in collection Iterator ()

Return an iterator used to loop over collection

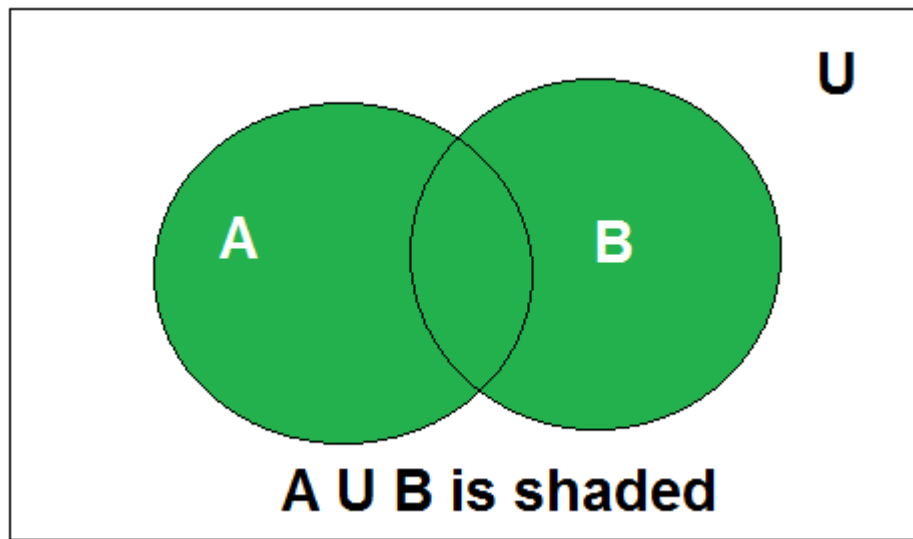
- e. Intersection of two sets
- f. Union of two sets
- g. Difference between two sets, h. Subset

2. **Objective** – To implement all set operations and represent them by creating an ADT.

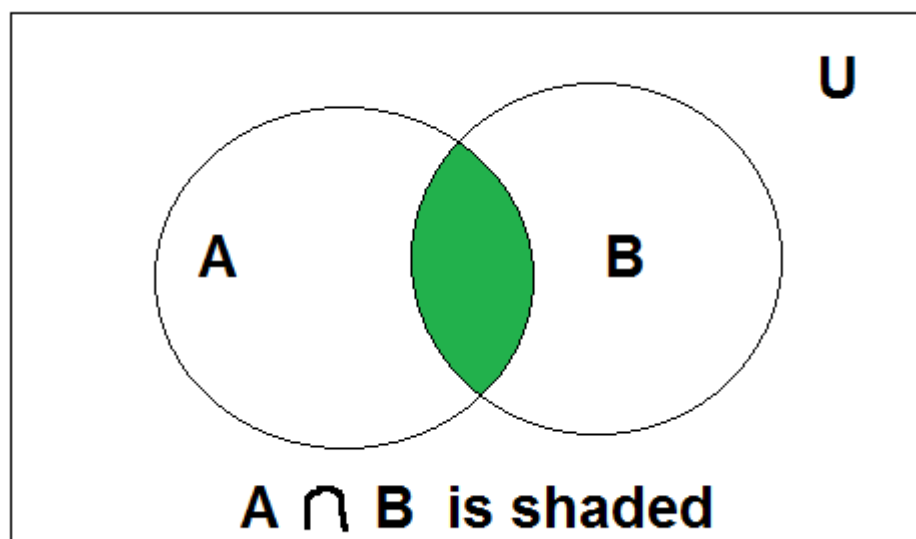
3. **Theory** - In [computer science](#), a **set** is an [abstract data type](#) that can store unique values, without any particular [order](#). It is a computer implementation of the [mathematical](#) concept of a [finite set](#). Unlike most other [collection](#) types, rather than retrieving a specific element from a set, one typically tests a value for membership in a set.

Sets can be implemented using various [data structures](#), which provide different time and space trade-offs for various operations. Some implementations are designed to improve the efficiency of very specialized operations, such as `nearest` or `union`. Implementations described as "general use" typically strive to optimize the `element_of`, `add`, and `delete` operations. A simple implementation is to use a [list](#), ignoring the order of the elements and taking care to avoid repeated values. This is simple but inefficient, as operations like set membership or element deletion are $O(n)$, as they require scanning the entire list.^[b] Sets are often instead implemented using more efficient data structures, particularly various flavors of [trees](#), [tries](#), or [hash tables](#).

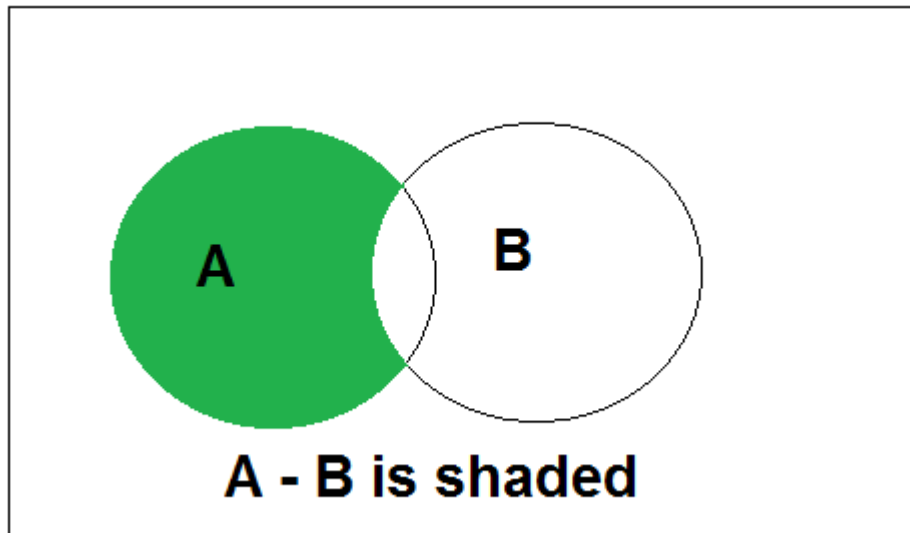
1. $\text{union}(S, T)$: returns the [union](#) of sets S and T .



2. $\text{intersection}(S, T)$: returns the [intersection](#) of sets S and T .



3. $\text{difference}(S, T)$: returns the [difference](#) of sets S and T .



4. $\text{subset}(S, T)$: predicate that tests whether the set S is a [subset](#) of set T
5. $\text{create}()$: creates a new, initially empty set structure.
6. $\text{add}(S, x)$: adds the element x to S , if it is not present already.
7. $\text{remove}(S, x)$: removes the element x from S , if it is present.
8. $\text{capacity}(S)$: returns the maximum number of values that S can hold.

4. Algorithm –

1. Intersection is an [associative](#) operation; that is, for any sets A , B , and C , one has $A \cap (B \cap C) = (A \cap B) \cap C$. Intersection is also [commutative](#); for any A and B , one has $A \cap B = B \cap A$. It thus makes sense to talk about intersections of multiple sets. The intersection of A , B , C , and D , for example, is unambiguously written $A \cap B \cap C \cap D$.
2. We say that A and B are *disjoint* if A does not intersect B . In plain language, they have no elements in common. A and B are disjoint if their intersection is [empty](#), denoted

3. .The union of two sets A and B is the set of elements which are in A , in B , or in both A and B . In symbols,

5. Program code –

```
#include<bits/stdc++.h>

#include <iostream>

using namespace std;

int set1[100],set2[100],no1,no2;

void insert()
{
    cout<<"Enter number of elements u want to insert in set
1"<<endl;

    cin>>no1;

    for(int i=0;i<no1;i++)

        cin>>set1[i];

    cout<<"Enter number of elements u want to insert in set
2"<<endl;

    cin>>no2;

    for(int i=0;i<no2;i++)

        cin>>set2[i];

}

void remove()
{
```

```
int num,i;
bool x=0;
cout<<"Enter a number u want to delete"<<endl;
cin>>num;
for(i=0;i<no1;i++)
{
    if(set1[i]==num)
    {
        x=1;
        break;
    }
}
while(i<(no1-1) && x==1)
{
    set1[i]=set1[i+1];
    i++;
}
if(x==1)
    no1--;
x=0;
for(i=0;i<no2;i++)
{
```

```
        if(set2[i]==num)
        {
            x=1;
            break;
        }
    }
    while(i<(no2-1) && x==1)
    {
        set2[i]=set2[i+1];
        i++;
    }
    if(x==1)
        no2--;
}

void search()
{
    int num,i;
    cout<<"Enter a number u want to search"<<endl;
    cin>>num;
    for(i=0;i<no1;i++)
    {
        if(set1[i]==num)
```

```
        {  
            cout<<"Number is present in set1"<<endl;  
            break;  
        }  
    }  
    if(i==no1)  
        cout<<"Number doesn't exist in set1"<<endl;  
    for(i=0;i<no2;i++)  
    {  
        if(set2[i]==num)  
        {  
            cout<<"Number is present in set2"<<endl;  
            break;  
        }  
    }  
    if(i==no2)  
        cout<<"Number doesn't exist in set2"<<endl;  
}  
void size()  
{  
    cout<<"Number of elements present in set 1 is  
"<<no1<<endl;
```

```
        cout<<"Number of elements present in set 2 is  
"<<no2<<endl;  
}  
void display()  
{  
    for(int i=0;i<no1;i++)  
        cout<<set1[i]<<"\t";  
    cout<<endl;  
    for(int i=0;i<no2;i++)  
        cout<<set2[i]<<"\t";  
    cout<<endl;  
}  
void intersection()  
{  
    int a[100],m=0,k=0;  
    for(int i=0;i<no1;i++)  
    {  
        for(int j=0;j<no2;j++)  
        {  
            if(set1[i]==set2[j])  
            {  
                a[k]=set1[i];
```



```
                k++;  
                m++;  
                break;  
            }  
        }  
    }  
    for(int i=0;i<m;i++)  
        cout<<a[i]<<"\t";  
    cout<<endl;  
}  
void uni(int x)  
{  
    int a[100],max=set1[0];  
    memset(a,0,sizeof(a));  
    for(int i=0;i<no1;i++)  
    {  
        a[set1[i]]=1;  
        if(max<set1[i])  
            max=set1[i];  
    }  
    for(int i=0;i<no2;i++)  
    {
```

```
        a[set2[i]]=1;
        if(max<set2[i])
            max=set2[i];
    }
    if(x==0)
    {
        for(int i=0;i<=max;i++)
        {
            if(a[i]==1)
                cout<<i<<"\t";
        }
        cout<<endl;
    }
    if(x==1)
    {
        for(int i=0;i<no2;i++)
        {
            if(a[set2[i]]==1)
                a[set2[i]]=0;
        }
        for(int i=0;i<=max;i++)
        {
```

```
                if(a[i]==1)
                    cout<<i<<"\t";
            }
            cout<<endl;
        }
    }
    void difference()
    {
        uni(1);
    }
    void subset()
    {
        int a[100],i;
        memset(a,0,sizeof(a));
        for(i=0;i<no1;i++)
        {
            a[set1[i]]=1;
        }
        for(i=0;i<no2;i++)
        {
            if(a[set2[i]]==0)
            {
```

```
        cout<<"Set 2 in not a subset of Set 1"<<endl;
        break;
    }
}
if(i==no2)
    cout<<"Set 2 is subset of Set 1"<<endl;
}
int main()
{
    int ch;
    bool f=1;
    while(f)
    {
        cout<<"1.Insert 2.Remove 3.Search 4.Size 5.Display
6.Intersection 7.Union 8.Difference 9.Subset"<<endl;
        cin>>ch;
        switch(ch)
        {
            case 1:
                insert();
                break;
            case 2:
```

```
        remove();  
        break;  
case 3:  
        search();  
        break;  
case 4:  
        size();  
        break;  
case 5:  
        display();  
        break;  
case 6:  
        intersection();  
        break;  
case 7:  
        uni(0);  
        break;  
case 8:  
        difference();  
        break;  
case 9:  
        subset();
```

```
        break;
    default:
        cout<<"Wrong choice"<<endl;
    }
    cout<<"If u want to continue then press 1 else
0"<<endl;
    cin>>f;
}
return 0;
}
```

6. Output screen shots/Copy Past From Terminal –

The first screenshot shows the C++ code for a set program. The code includes `<bits/stdc++.h>` and `<iostream>`, uses the `std` namespace, and defines two sets, `set1` and `set2`, each of size 100. The `insert` function is defined to take a set, an element, and a flag to indicate if the element should be inserted. The main function displays a menu with options 1 to 9. The user enters 1 to insert elements into set 1. The program prompts for the number of elements to insert (5) and the elements themselves (4, 5, 7, 8). The second screenshot shows the user entering 4 to delete an element from set 1. The program prompts for the element to delete (4). The final output shows the number of elements in set 1 is 2 and the number of elements in set 2 is 3.

```

1 #include<bits/stdc++.h>
2 #include <iostream>
3 using namespace std;
4 int set1[100],set2[100],no1,no2;
5 void insert()
6 {
7     1.Insert 2.Remove 3.Search 4.Size 5.Display 6.Intersection 7.Union 8.Difference 9.Subset
8     Enter number of elements u want to insert in set 1
9
10    Enter number of elements u want to insert in set 2
11    }
12    }
13    }
14    }
15    }
16    }
17    }
18    }
19    if u want to continue then press 1 else 0
20    1.Insert 2.Remove 3.Search 4.Size 5.Display 6.Intersection 7.Union 8.Difference 9.Subset
21    5 7
22    8 7
23    }
24    if u want to continue then press 1 else 0
25    1.Insert 2.Remove 3.Search 4.Size 5.Display 6.Intersection 7.Union 8.Difference 9.Subset
26    0
27    }
28    if u want to continue then press 1 else 0
29    1.Insert 2.Remove 3.Search 4.Size 5.Display 6.Intersection 7.Union 8.Difference 9.Subset
30    4 5 7 8
31    }
32    if u want to continue then press 1 else 0
33    }
34    }
35    if(x==1)
36    {
37        no1--;
38        x=0;
39    }
40    }
41    }
42    }
43    }
44    }
45    }
46    }
47    }
48    }
49    }
50    }
51    }
52    }
53    }
54    }
55    }
56    }
57    }
58    }
59    }
60    }
61    }
62    }
63    }
64    }
65    }
66    }
67    }
68    }
69    }
70    }
71    }
72    }
73    }
74    }
75    }
76    }
77    }
78    }
79    }
80    }
81    }
82    }
83    }
84    }
85    }
86    }
87    }
88    }
89    }
90    }
91    }
92    }
93    }
94    }
95    }
96    }
97    }
98    }
99    }
100   }

```

Compilation results...

```

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\hp\Documents\SD\assign1.exe
- Output Size: 1.83584785461426 MiB
- Compilation Time: 0.66s

```

Line: 9 Col: 27 Set: 0 Lines: 221 Length: 3367 Insert Done parsing in 0.015 seconds

The second screenshot shows the same code, but with the user entering 4 to delete an element from set 1. The program prompts for the element to delete (4). The final output shows the number of elements in set 1 is 2 and the number of elements in set 2 is 3.

```

1 #include<bits/stdc++.h>
2 #include <iostream>
3 using namespace std;
4 int set1[100],set2[100],no1,no2;
5 void insert()
6 {
7     1.Insert 2.Remove 3.Search 4.Size 5.Display 6.Intersection 7.Union 8.Difference 9.Subset
8     Enter number of elements u want to insert in set 1
9
10    Enter number of elements u want to insert in set 2
11    }
12    }
13    }
14    }
15    }
16    }
17    }
18    }
19    if u want to continue then press 1 else 0
20    1.Insert 2.Remove 3.Search 4.Size 5.Display 6.Intersection 7.Union 8.Difference 9.Subset
21    5 7
22    8 7
23    }
24    if u want to continue then press 1 else 0
25    1.Insert 2.Remove 3.Search 4.Size 5.Display 6.Intersection 7.Union 8.Difference 9.Subset
26    0
27    }
28    if u want to continue then press 1 else 0
29    1.Insert 2.Remove 3.Search 4.Size 5.Display 6.Intersection 7.Union 8.Difference 9.Subset
30    4 5 7 8
31    }
32    if u want to continue then press 1 else 0
33    }
34    }
35    if(x==1)
36    {
37        no1--;
38        x=0;
39    }
40    }
41    }
42    }
43    }
44    }
45    }
46    }
47    }
48    }
49    }
50    }
51    }
52    }
53    }
54    }
55    }
56    }
57    }
58    }
59    }
60    }
61    }
62    }
63    }
64    }
65    }
66    }
67    }
68    }
69    }
70    }
71    }
72    }
73    }
74    }
75    }
76    }
77    }
78    }
79    }
80    }
81    }
82    }
83    }
84    }
85    }
86    }
87    }
88    }
89    }
90    }
91    }
92    }
93    }
94    }
95    }
96    }
97    }
98    }
99    }
100   }

```

Compilation results...

```

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\hp\Documents\SD\assign1.exe
- Output Size: 1.83584785461426 MiB
- Compilation Time: 0.66s

```

Line: 9 Col: 27 Set: 0 Lines: 221 Length: 3367 Insert Done parsing in 0.015 seconds

7. Conclusion –

The importance of sets is one. They allow us to treat a *collection of mathematical objects* as a mathematical object on its own right.

They are **important** everywhere in mathematics because every field of mathematics uses or refers to **sets** in some way. They are **important** for building more complex mathematical structure