Objective:

- It will help in understanding the use of array of objects.
- Handling memory issues related to struct objects on heap.

Task-1

Must complete it before lab-2 and put it in your account (Z drive) before coming to lab.

It is the Set database application that we discussed today. The function prototypes are quite self explaining and basic skeleton of application is also written in setDatabase() function in SetDatabase.cpp.

Set.h

```
struct Set
{
    int * data;
    int noOfElements;
    int capacity;
    char name[10];
};
void createSet ( Set &, char * name, int n );
bool addElement ( Set &, int element );
bool removeElement ( Set &, int element );
bool searchElement ( Set, int element );
int searchElementPosition ( Set, int element );
bool isEmpty( Set );
bool isFull( Set );
void displaySet ( Set );
Set intersection ( Set, Set );
Set calcUnion ( Set, Set );
Set difference ( Set, Set );
int isSubset ( Set, Set );
void reSize( Set &, int newSize );
void displayPowerSet ( Set );
Set creatClone ( Set & source );
void deallocateSet ( Set & );
```

Set.cpp

```
void deallocateSet ( Set & s )
{
    if ( s.data )
    {
        delete [] s.data;
    }
    s.data=0;
    s.noOfElements = 0;
    s.capacity = 0;
}
void createSet ( Set & s, char * setName, int n)
{
    strcpy(s.name, setName);
    s.capacity = n;
    s.noOfElements = 0;
    if (s.capacity==0)
        s.data=0;
    else
        s.data = new int[s.noOfElements];
}
Set createClone( Set & s)
{
```

```
    Set x;
    x.noOfElements = s.noOfElements;
    strcpy(x.name, s.name);
    x.capacity = s.capacity;
    if (x.capacity==0)
        x.data=0;
    else
    {
        x.data = new int [x.capacity];
        for (int i=0; i<x.noOfElements; i++)
        {
            x.data[i] = s.data[i];
        }
    }
    return x;
}
int searchElementPosition ( Set s, int element )
{
    int i=0;
    while ( i < s.noOfElements && s.data[i] != element)
    {
        i++;
    }
    int x =  i != s.noOfElements? i : -1;
    deallocateSet(s);
    return x;
}

bool addElement ( Set & s, int element )
{
    int pos = searchElementPosition ( createClone(s), element );
    if ( pos != -1 )
        return false;
    s.data[s.noOfElements] = element;
    s.noOfElements = s.noOfElements + 1;
    return true;
}
bool removeElement ( Set & s, int element )
{
    int pos = searchElementPosition ( createClone(s), element );
    if ( pos != -1 )
        return false;
    s.data[pos] = s.data[s.noOfElements-1];
    s.noOfElements = s.noOfElements - 1;
    return true;
}
void displaySet ( Set s)
{
    cout<<s.name<<" = {";
    for ( int i=0; i < s.noOfElements; i++ )
    {
        cout<<s.data[i]<<", ";
    }
    if (s.noOfElements==0)
        cout<<"}";
    else
        cout<<"}";//"\b\b}";
    deallocateSet(s);
}
int getCommonElments(Set a, Set b)
{
    int count=0;
```

```
    for ( int i=0; i<a.noOfElements; i++)
    {
        for ( int j=0; j<b.noOfElements; j++)
        {
            if (a.data[i]==b.data[j])
                count++;
        }
    }
    deallocateSet(a);
    deallocateSet(b);
    return count;
}
Set calUnion( Set a, Set b)
{
    int commonElements = getCommonElments(createClone(a),
                                          createClone(b));
    int unionSize = a.noOfElements + b.noOfElements - commonElements;
    Set un;
    createSet(un, " ", unionSize);

    for ( int i=0; i<a.noOfElements; i++)
    {
        un.data[i] = a.data[i];
        un.noOfElements = un.noOfElements+1;
    }
    for ( int j=0; j<b.noOfElements; j++)
    {
        addElement(un, b.data[j]);
    }
    deallocateSet(a);
    deallocateSet(b);
    return un;
}
```

SetDatabase.h

```
struct SetDatabase
{
    Set * sets;
    int noOfSets;
    int capacity;
};
void createSetDatabase( SetDatabase & setDB, int size);
int findSetPositionInDatabase(SetDatabase & setDB, char * setName);
bool addSetInDatabase( SetDatabase & setDB, Set a);
bool addSetInDatabase( SetDatabase & setDB, char * setName="A", int size=5);
bool removeSetInDatabase(SetDatabase & setDB, char * setName);
void displayAllSets (SetDatabase & setDB);
bool addElementInSetInSetDatabase(SetDatabase & setDB, char * setName, int ele);
bool removeElementInSetInSetDatabase(SetDatabase & setDB, char * setName, int ele);
Set calUnionInSetDatabase(SetDatabase & setDB, char * setA, char * setB);
void setsDatabase();
```

SetDatabase.cpp

```
void createSetDatabase( SetDatabase & setDB, int size)
{
    setDB.noOfSets = 0;
    setDB.capacity = size;
    setDB.sets = new Set[setDB.capacity];
}
void setsDatabase()
```

```
{
    SetDatabase setDB;
    createSetDatabase(setDB, 10);
    int choice;
    char name[10];
    int size;
    do
    {
        cout<<"\n           Set Collections\n";
        cout<<"=======================================\n";
        cout<<"Enter  1. to add a Set\n"
            <<"       2. to remove a Set\n"
            <<"       3. display all sets\n"
            <<"       4. to display a Set\n"
            <<"       5. to add an element in Set\n"
            <<"       6. to remove an element from Set\n"
            <<"       7. to calculate Union\n"
            <<"       8. to calculate Intersection\n"
            <<"       9. to calculate Difference\n"
            <<"      10. to find subset\n"
            <<"      11. to display Power Set\n"
            <<"       0. to exit Application\t\t";
        cin>>choice;
        if (choice==1)
        {
            cout<<"\nEnter Set Name to Add: ";
            cin>>name;
            if (findSetPositionInDatabase(setDB, name)!=-1)
                cout<<"\nDuplication Error: Already have same name set";
            else
            {
                cout<<"\nEnter The Size of Set: ";
                cin>>size;
                if (addSetInDatabase(setDB,name,size))
                    cout<<"\nSet Added Successfully";
                else
                    cout<<"\nUnable to Add Set";
            }
        }
        else if ( choice == 3)
        {
            displayAllSets(setDB);
        }
        else if ( choice == 5 )
        {
            cout<<"\nEnter Set Name to Add Element: ";
            cin>>name;
            if (findSetPositionInDatabase(setDB, name)!=-1)
            {
                int ele;
                cout<<"\nEnter Element to Add: ";
                cin>>ele;
                if (addElementInSetInSetDatabase(setDB, name, ele))
                    cout<<"\nElement Added Successfully";
                else
                    cout<<"\nUnable to Add Element";
            }
            else
                cout<<"\n Set not Found";

        }
        else if (choice==7)
```

```
        {
            char setA[10], setB[10];
            cout<<"\nEnter Name of First Set";
            cin>>setA;
            cout<<"\nEnter Name of Second Set";
            cin>>setB;
            Set un = calUnionInSetDatabase(setDB, setA, setB);
            displaySet(createClone(un));
            char ch;
            cout<<"\nDo you want to store it in database [y/n]";
            cin>>ch;
            if (ch=='y')
            {
                bool flag;
                do
                {
                    flag=false;
                    cout<<"\nEnter Set Name";
                    cin>>name;
                    if (findSetPositionInDatabase(setDB, name)!=-1)
                    {
                        cout<<"\nduplication of Name";
                        flag=true;
                    }
                }
                while(flag);
                strcpy(un.name,name);
                addSetInDatabase(setDB, un);
            }
            else
            {
                deallocateSet(un);
            }

        }
    }
    while(1);

}
```

Driver.cpp

```
int main()
{
    setsDatabase();

    return 1;
}
```

Sample Object Layout

```
SetDatabase setDB;
```