Objective:

- Learn to write Function Templates and Class Templates.
- Both tasks will help you understand the importance of C++ templates and you will get a chance to use a new feature of C++11 standard i.e. initializer_list and range based for loop.

Global Functions

Task – 1:

In class, we developed the function 'mySwap(T &, T &)' using aliases to swap the contents of the received entities.
Now develop the same function again but using pointers.

Task – 2:

Write a function template named 'myGenericSort', which receives an array of size N and rearranges the elements of the array in ascending order.
Test your code by passing arrays of different types (primitive and user define).

Classes

Task – 3:

Complete Generic Array Class

```
#include <iostream>
#include <initializer_list>
using  namespace std;

template <typename T>
class Array
{
public:
    T *data;
    int size;
    bool isValidIndex(int index) const
public:
    Array();
    Array& operator = (const Array & ref);
    Array(const Array & ref);
    Array(Array && ref);
    Array& operator = (Array && ref);
    ~Array();
    const T & operator [] (int index) const;
    T & operator [] (int index);
    void reSize(int s);
    int getSize() const;
    //Array(initializer_list<T> list);//={});
    //Note: C language feature; variable number of arguments has its own importance
};
```

Sample Run:

```
int main()
{
    cout<<"\n2D Array using initializer list\n";
    Array< Array<int> > b= { {1,2,3},{4,5,6} };
    for ( int i=0; i<b.getSize(); i++ )
    {
        for ( int j=0; j<b[i].getSize(); j++ )
        {
            cout<<b[i][j]<<"\t";
        }
        cout<<"\n";
    }
    cout<<"\n3D Array using initializer list\n";
    Array< Array < Array<int> > > a= { { {1,2,3},{4,5,6} },{ {7,8,9},{10,11,12} } };
    for ( int i=0; i<a.getSize(); i++ )
    {
```

```
        for ( int j=0; j<a[i].getSize(); j++ )
        {
            for ( int k=0; k<a[i][j].getSize(); k++ )
            {
                cout<<a[i][j][k]<<"\t";
            }
            cout<<"\n";
        }
    }
    cout<<"\n3D Array using resize method\n";
    Array< Array < Array< int > > > c;
    c.reSize(2);
    int val=1;
    for ( int i=0; i<2; i++)
        c[i].reSize(3);
    for ( int i=0; i<c.getSize(); i++ )
    {
        for ( int j=0; j<c[i].getSize(); j++ )
        {
            c[i][j].reSize(4);
        }
    }
    for ( int i=0; i<c.getSize(); i++ )
    {
        for ( int j=0; j<c[i].getSize(); j++ )
        {
            for ( int k=0; k<c[i][j].getSize(); k++ )
            {
                c[i][j][k]=val;
                val++;
            }
        }
    }
    for ( int i=0; i<c.getSize(); i++ )
    {
        for ( int j=0; j<c[i].getSize(); j++ )
        {
            for ( int k=0; k<c[i][j].getSize(); k++ )
            {
                cout<<c[i][j][k]<<"\t";
            }
            cout<<"\n";
        }
        cout<<"\n";
    }
    return 0;
}
```

Sample Output:

```
2D Array using initializer list
1       2       3
4       5       6

3D Array using initializer list
1       2       3
4       5       6
7       8       9
10      11      12

3D Array using resize method
1       2       3       4
5       6       7       8
9       10      11      12

13      14      15      16
17      18      19      20
21      22      23      24
```

Task – 4:

First complete Task-3

Develop generic Matrix ADT

It must support all the operations that were given in the latest version of Matrix class developed in OOP class/lab.

You must test your code by creating and using the following objects

```
Matrix < CString > m1(2,3);
Matrix < Matrix < int > > m1(3,4);
Matrix < Array < int > > m2(3,4);
Matrix < Matrix < String > > m3(3,4);
```

For Next Lecture

Task – 0:

Some basic usage of Initializer list and Range based For loop and auto keyword.

This type is used to access the values in a C++ initialization list, which is a list of elements of type const T.

Objects of this type are automatically constructed by the compiler from initialization list declarations, which is a list of comma-separated elements enclosed in braces:

```
auto il = { 10, 20, 30 };  // the type of il is an initializer_list
```

```
int main()
{
    int arr[5] = {10,20,30,40,50};
    for ( auto val : arr)
    {
        cout<<val<<"\t";
    }
    initializer_list<int> a = {1,2,3,4,5};
    cout<<"\n";
    for (int val : a)
    {
        cout<<val<<"\t";
    }
    cout<<"\n";

    const int * val = a.begin();
    for (int i=1; i<=a.size(); i++)
    {
        cout<<*val<<"\t";
        val++;
    }

    cout<<"\n";
    a = {13, 15, 17};
    for (int val : a)
    {
        cout<<val<<"\t";
    }
    cout<<"\n";

    auto a3 = {1,2,3,4,5};//its type is also initializer_list<int>
    cout<<endl<<typeid(a3).name()<<endl;

    cout<<"\n";
    for (auto it = rbegin(arr); it != rend(arr); ++it)
        cout << *it << '\t';

    cout<<"\n";
    for (auto it = rbegin(a); it != rend(a); ++it)
        cout << *it << '\t';

            cout<<'\n';
```

```
    return 0;
}
```

Code Output

```
10   20   30   40   50
1    2    3    4    5
1    2    3    4    5
13   15   17

St16initializer_listIiE

50   40   30   20   10
17   15   13
Program ended with exit code: 0
```

Task – 1:

The following 'Set' class declaration is designed for Set of integers.

In this task, you are required to convert the following class into Generic 'Set' class/Class-Template.

While implementing the Class Template 'Set', if you need to use array then you are only allowed to use Generic Array class implemented in last lecture.

You are free to add or subtract or modify the data members for the Generic Set class but you must have a solid reason for every change.

```
class Set
{
    int * data;
    int capacity;
    int noOfElements;
public:
    Set( int cap = 5 );
    ~Set();
    void insert (int element);
    void remove (int element);
    void print() const;
    int getCardinality() const;
    bool isMember ( int val ) const;
    int isSubSet ( const Set & s2 ) const;
    void reSize ( int newcapacity );
    Set( const Set & ref);
    Set calcUnion ( const Set & s2 ) const;
    Set calcIntersection ( const Set & s2 ) const;
    Set calcDifference ( const Set & s2 ) const;
    Set calcSymmetricDifference ( const Set & s2 ) const;
};
```