# Objectives

After performing this lab, students shall be able to implement:

* Template classes
* Template Functions

**TASK-1: Template Classes**

Consider the following class template:

template <class T, int N>

class Sequence {

T memblock [N];

public:

void setmember (int x, T value);

T getmember (int x);

};

Sequence is a class that stores a sequence of elements. N is an integer. The member function setmember sets the member at position x in the memblock with value and getmember returns the value at index x.

1. Implement the Sequence class w.r.t the following **main**

int main ()

{

Sequence <int,5> myints;

Sequence <double,5> myfloats;

myints.setmember (0,100);

myfloats.setmember (3,3.1416);

cout << myints.getmember(0) << '\n';

cout << myfloats.getmember(3) << '\n';

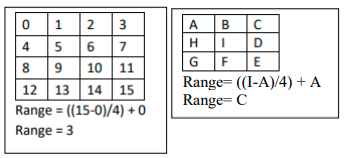
return 0;

}

**Task 2:**

You have to design a C++ **template** function **range**, which takes a dynamic two-dimensional square matrix, its dimensions (rows, columns) size. It returns the range of values in matrix.

* Range = ((max – min)/4) + min.
* **Note:** No specialization is required for this function. **Do not** take **input** from **user**. **Initialize** a 2D array in **main** and call the function.
* **Template <typename T> T range (T \*\*array, int rows, int columns)**



Write another template function that will Shift the columns of matrix by 1 and print the shifted matrix. What would be the prototype of this function?

|  |  |  |
| --- | --- | --- |
| C | A | B |
| D | H | I |
| E | G | F |

**Task 3:**

**Give implementation of class template “Vector”. A vector is a container which contains elements in a dynamic array. Following are the required functionalities for your vector:**

1. A member called “arrayPtr” of type T\* which contains the array.
2. A member of int type called “capacity” which stores the information about total number of elements that can be stored in the vector. iii. A member of int type called “totalElements” which stores the information about actual number of elements that have been stored in the array.
3. An overloaded constructor which takes capacity as parameter, and creates an array of given capacity.
4. A copy constructor which deep copies another object of Vector.
5. A method called insertElement(T const& param) which stores the element in the next available position (position could be determined by “totalElements” member) of array. When the array becomes full, the insertElement method first doubles the capacity of array and then inserts the element.
6. A method that overloads stream operator “<<”.
7. Implement getters for getting the values of totalElements and capacity.

**Now run the following main program:** int main()

{

Vector< int > iv1(2); iv1.insertElement(5); iv1.insertElement(6); iv1.insertElement(1);

Vector< int > iv2(iv1); cout<<iv2;

}

**Task-4: Exception Handling**

Write a program that prompts the user to enter a person’s date of birth in numeric form such as 8-27-1980. The program then outputs the date of birth in the form: August 27, 1980. Your program must contain three exception classes: **invalidDay, invalidMonth,** and **invalidYear**. If the user enters an invalid value for day, then the program should throw and catch an **invalidDay** object. Follow similar convention for the invalid values of month. Handle leap year value with **invalidYear** exception.