CSE225L – Data Structures and Algorithms Lab
Lab 03
Template Class and Operator Overloading

Task 1: Recall the class we used in the previous lab to allocate memory dynamically. Modify the header file and the source file given below so that they now work as template class (the array elements in the dynamically allocated memory can be any type as the user defines).

dynarr.h

```
#ifndef DYNARR_H_INCLUDED
#define DYNARR_H_INCLUDED

class dynArr
{
    private:

        int *data;
        int size;

    public:
       dynArr(int);
       ~dynArr();
       void setValue(int, int);
       int getValue(int);
};

#endif // DYNARR_H_INCLUDED
```

dynarr.cpp

```
#include "dynarr.h"
#include <iostream>
using namespace std;

dynArr::dynArr(int s)
{
    data = new int[s];
    size = s;
}
dynArr::~dynArr()
{
    delete [] data;
}
int dynArr::getValue(int index)
{
    return data[index];
}
void dynArr::setValue(int index, int value)
{
    data[index] = value;
}
```

Task 2: Recall the complex number class we discussed in our lectures. Modify the class and overload the ***** (multiplication) and the **!=** (not equal) operators for the class given below.

complex.h

```
#ifndef COMPLEX_H_INCLUDED
#define COMPLEX_H_INCLUDED

class Complex
{
    public:
    Complex();
    Complex(double, double);
    Complex operator+(Complex);
    void Print();
    private:
    double Real, Imaginary;
};

#endif // COMPLEX_H_INCLUDED
```

complex.cpp

```
#include "complex.h"
#include <iostream>
using namespace std;

Complex::Complex()
{
    Real = 0;
    Imaginary = 0;
}
Complex::Complex(double r, double i)
{
    Real = r;
    Imaginary = i;
}
Complex Complex::operator+(Complex a)
{
    Complex t;
    t.Real = Real + a.Real;
    t.Imaginary = Imaginary + a.Imaginary;
    return t;
}
void Complex::Print()
{
    cout << Real << endl;
    cout << Imaginary << endl;
}
```