

**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 of any type).

<pre><b>dynarr.h</b>  #ifndef DYNARR_H_INCLUDED #define DYNARR_H_INCLUDED  <b>class</b> dynArr {     private:          int *data;         int size;      public:         dynArr(int);         ~dynArr();         void setValue(int, int);         int getValue(int); };  #endif // DYNARR_H_INCLUDED</pre>	<pre><b>dynarr.cpp</b>  #include "dynarr.h" #include &lt;iostream&gt; using namespace std;  <b>dynArr::dynArr</b>(int s) {     data = new int[s];     size = s; }  <b>dynArr::~~dynArr</b>() {     delete [] data; }  <b>int dynArr::getValue</b>(int index) {     return data[index]; }  <b>void dynArr::setValue</b>(int index, int value) {     data[index] = value; }</pre>
--	---

**Task 2:** Recall the complex number class we discussed in our lectures. Modify the class and overload the \* (multiplication) and the == (equal) operators for the class given below.

<pre><b>complex.h</b>  #ifndef COMPLEX_H_INCLUDED #define COMPLEX_H_INCLUDED  <b>class</b> Complex {     public:         Complex();         Complex(double, double);         Complex operator+(Complex);         void Print();      private:         double Real, Imaginary; };  #endif // COMPLEX_H_INCLUDED</pre>	<pre><b>complex.cpp</b>  #include "complex.h" #include &lt;iostream&gt; using namespace std;  <b>Complex::Complex</b>() {     Real = 0;     Imaginary = 0; }  <b>Complex::Complex</b>(double r, double i) {     Real = r;     Imaginary = i; }  <b>Complex Complex::operator+</b>(Complex a) {     Complex t;     t.Real = this-&gt;Real + a.Real;     t.Imaginary = this-&gt;Imaginary + a.Imaginary;     return t; }  <b>void Complex::Print</b>() {     cout &lt;&lt; Real &lt;&lt; endl;     cout &lt;&lt; Imaginary &lt;&lt; endl; }</pre>
---	---