CSE225L: Data Structures and Algorithm Lab

Lab 04: Class Template

North South University

Task 1. 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

using namespace std;
class dynArr {
private:
      int *data,size;
public:
      dynArr(int);
      ~dynArr();
      Void setValue(int,int);
      int getValue(int);
};
#include "dynarr.tpp"
#endif // DYNARR_H_INCLUDED
```

dynarr.tpp

```
#include "dynarr.h"
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;
}
int dynArr::getSize(){
      return size;
}
```

Task 2: Convert the following class to class template.

Complex.h

```
#ifndef COMPLEX_H_INCLUDED
#define COMPLEX_H_INCLUDED

using namespace std;
class Complex{
public:
    Complex();
    Complex(int, int);
    Complex operator+(Complex);
    void Print();
private:
    int Real, Imaginary;
};

#include "Complex.tpp"
#endif // COMPLEX_H_INCLUDED
```

Complex.tpp

```
#include "Complex.h"
Complex::Complex(){
    Real = 0;
    Imaginary = 0;
}
Complex::Complex(int r, int 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(){
    if (Real==0)
        cout << Imaginary<<"i"<<endl;
    else{
       if(Imaginary<0)
         cout<<Real<<Imaginary<<"i"<<endl;
       else if(Imaginary==0)
         cout<<Real<<endl;
       else
         cout<<Real<<"+"<<Imaginary<<"i"<<endl;
    }
}
```