Object Oriented Programming – Fall 22

(BS-CS-F22)

Lab-8

Lab Instructor: Maa'm Sanam Ahmad

Instructions:

- **Indent your code properly.**
- **Use meaningful variable and function names.**
- **Use the camelCase notation.**
- **Use meaningful prompt lines/labels for all input/output.**
- **Do NOT use any GLOBAL variable(s). However, global named constants may be used.**
- **This is an individual lab, you are strictly NOT allowed to discuss your solution with fellow colleagues, even not allowed to ask how is he/she is doing, it may result in negative marking. You can ONLY discuss with your TAs or with me. • Anyone caught in an act of plagiarism wouldbe awarded an "F" grade in this Lab.**
- **Do Validations on inputs where required otherwise 1 mark will be deducted for every wrong validation.**

TASK-1:

Create a class IntArray that has 2 private member variables to store following information:

• aptr (an integer array pointer)

• arraySize (an integer)

Note: All member functions of the IntArray class, which are not supposed to change data stored in the calling object, should be declared as const member functions.

a) Implement overloaded constructor that takes size s (Initialize arraySize to s and all elements of aptr to 0) , destructors, getter and setter functions for member variables.(5)

b) Overload the **int &operator[](int index)** operator to allow access to elements of the dynamic array. First check if index is out of bound then return a default value -999. Otherwise, return element of array of that index. (5)

c) Overload the += operator for IntArray class which can be used to concatenate the current IntArray object with another IntArray object. Your function should allocate a new array (of an appropriate size) and should successfully perform the concatenation.(10)

d) Your task is to overload the assignment operator (=) for class IntArray. Overloading the assignment operator allows objects to be assigned using the standard assignment syntax.(5)

TASK-2

With the continuation of Task1 also overload following operators:

```
class Complex {
private:
        double real;
        double imaginary;
}
```

a) Implement below operators for class Complex.

```
Complex operator+(const Complex &c) const; // Addition
Complex operator-(const Complex &c) const; // Subtraction
Complex operator*(const Complex &c) const; // Multiplication
Complex operator/(const Complex &c) const;
```