NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD CAMPUS

CS-1004 Object Oriented Programming Fall-2021

ASSIGNMENT-03

Section (All)

Submission Deadline: 14 November, 2021 11:59 pm

Instructions:

1. This is an individual assignment. You must complete this assignment by yourself. You cannot work with anyone else in the class or with someone outside of the class. The code you write must be your own and you must understand each part of your code. You are encouraged to get help from the instructional staff through Google Classroom.
2. Do not use any String or Math libraries (such as cmath, cstring, string, etc.) and also do not use built-in functions (such as pow, etc.). **Caution**: zero marks will be awarded.
3. **Test cases:** Test cases are not mandatory; however, test cases file (in GTest) will be shared with you on Google Classroom. Test cases have bonus points other than 100 marks of assignment.
4. **Driver:** Write a separate *main()* function as a driver for each question. The driver should demonstrate all the required functionalities of a question.
5. **Source Code:** Your code must be *generic**,** well-commented**,*** and *well-organized*. It should not have compilation errors. Each class should have private data members. Each class should also contain constructors (*default*, *parametrized*, and *copy constructor* if needed), a destructor, setters, and getters. Each class should be implemented in separate .h and .cpp files.
6. **Marks Distribution:** Total marks of the assignment are 100. There are two questions. Question #1 carries 60 marks and Question #2 carries 40 marks. Marks distribution of each part is mentioned corresponding to that part.
7. **Submission Guidelines**: Submit assignment according to the guidelines given below.
    a. Combine all your work in one .zip file. Use a proper naming convention for your submission file. Name the .zip file as ROLL-NUM_SECTION (e.g., 20i-0001_A). Your zip file should contain separate folders for each question, but no subfolders within each folder. Each folder should contain the .h and .cpp files having necessary (#ifndef) header guards.
    b. Run and check your program on a lab machine before submission. If there is a syntax error, zero marks will be awarded in that specific question.
    c. Submit the .zip file on Google Classroom within the deadline.
    d. Submission other than Google classroom (e.g., email, etc.) will not be accepted. If we cannot download the file from Google classroom due to any reason it will lead to zero marks in the assignment.
8. **Plagiarism**: Plagiarism of any kind (copying from others, copying from the internet, etc.) is not allowed. If found plagiarized, you will be awarded zero marks in the assignment. Repeating such an act can lead to strict disciplinary actions and failure in the course.

Note: Make sure that you have read and understood every instruction before starting the assignment.

Q1: Operator Overloading for Complex Numbers [Marks: 60]

Develop a class named *Complex* for representing complex numbers. A complex number has the general form *a + bi*, where *a* is the real part and *b* is the imaginary part (*i* stands for imaginary). Arithmetic operations on complex numbers are as follows:

$(a + bi) + (c + di) = (a + c) + (b + d)\,i$

$(a + bi) - (c + di) = (a - c) + (b - d)\,i$

$(a + bi) * (c + di) = (ac - bd) + (ad + bc)\,i$

$(a + bi) / (c + di) = [(ac + bd) + (bc - ad)\,i] / (c^2 + d^2)$

The class should also provide the following overloaded operator capabilities:

a. Overload the addition operator (+) to add two Complex numbers. [Marks: 4]

b. Overload the subtraction operator (–) to subtract two Complex numbers. [Marks: 4]

c. Overload the multiplication operator (*) to multiply two Complex numbers. [Marks: 4]

d. Overload the division operator (/) to divide two Complex numbers. [Marks: 4]

e. Overload the addition assignment operator (+=), subtraction assignment operator (–=), multiplication assignment operator (*=), and division assignment operator (/=). [Marks: 4+4+4+4=16]

f. Overload the assignment operator (=) to assign one Complex number to another. [Marks: 4]

g. Overload the stream insertion (<<) to get the input Complex numbers and stream extraction operator (>>) to display Complex numbers. [Marks: 4+4=8]

h. Overload the comparison operators (==) and (!=) to check whether two Complex numbers are equal or not. [Marks: 4+4=8]

i. Overload the comparison operators (<) and (>) to check whether one Complex number is less or greater than the other. [Marks: 4+4=8]

Define all overloaded operators (except for stream insertion '<<' and stream extraction '>>' operators) as member functions of the *Complex* class. The overloaded function for stream insertion (<<) and stream extraction (>>) operator should be *non-member friend* functions. You need to write three files (complex.h, complex.cpp, and complexMain.cpp).

Q2: Operator Overloading for Polynomial Class [Marks: 40]

Develop a class named *Polynomial*. The internal representation of a polynomial is an array of terms. Each term contains a coefficient and an exponent. The term

$$2x^4$$

has coefficient 2 and the exponent 4. For a third-degree polynomial

$$4x^3 + 3x + 2$$

there are two terms and a constant. The first term has coefficient 4 and the exponent 3, the second term has coefficient 3 and the exponent 1, and the third is a constant 2. The detailed guideline to add/subtract polynomials can be found here.

The class should also provide the following overloaded operator capabilities:

a. Overload the addition operator (+) to add two polynomials. [Marks: 5]
b. Overload the subtraction operator (–) to subtract two polynomials. [Marks: 5]
c. Overload the addition operator (+=) to add two polynomials [Marks: 5]
d. Overload the subtraction operator (–=) to subtract two polynomials. [Marks: 5]
e. Overload operator (=) to assign one polynomial to another. [Marks: 5]
f. Overload operator (==) to check if two polynomials are equal. [Marks: 5]
g. Overload the stream insertion (<<) to get input polynomial and stream extraction operator (>>) to display the polynomial. [Marks: 5+5=10]

Define all overloaded operators (except for stream insertion '<<' and stream extraction '>>' operators) as member functions of the *Polynomial* class. The overloaded function for stream insertion (<<) and stream extraction (>>) operator should be *non-member friend* functions. You need to write three files (polynomial.h, polynomial.cpp, and polynomialMain.cpp).