NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD
Object Oriented Programming (CS 217)
Spring 2020

ASSIGNMENT # 4

Due Date: April 6, 2020 (5:00 PM)
Instructions:

- Make sure that you read and understand each and every instruction. If you have any questions or comments you are encouraged to discuss with your colleagues and instructors on Piazza https://piazza.com/nu.edu.pk/spring2020/cs217/home).
- Create each problem solution in three separate files (ClassName.h, ClassName.cpp and main file for each question as Q1 as 'q1.cpp', Q2 as 'q2.cpp' and Q3 as 'q3.cpp'. Specific instructions are given with each question.
- In main file of each question (e.g. q1.cpp, q2.cpp etc.) you have to implement the code to test each functionality.
- Combine all your work in one .zip file.
- Name the .zip file as ROLL-NUM SECTION.zip (e.g. 19i-0001 B.zip).
- Submit the .zip file on Google Classroom within the deadline.
- Start early otherwise you will struggle with the assignment.
- You must follow the submission instructions to the letter, as failing to do so will get you a zero in the assignment.
- All the submitted evaluation instruments (quizzes, assignments, lab work, exams, and the project) will be checked for plagiarism. If found plagiarized, both the involved parties will be marked.

Q 1: Operator overloading for Fraction Class - Develop a class named Fraction. This class handles fraction's numerator and denumerator. It should include three constructors. One with no parameters and which creates the fraction 0/1, one with one parameter numer and which creates the fraction numer/1, and one with two parameters and which creates the fraction numer/denom, but which assures that the fraction will be in normalized form (that is, positive denominator and with the greatest common divisor removed from the numerator and denominator). Make sure that the store method also stores fractions in normalized form. The class should also have copy constructor. Add three public methods (functions) to the class, called getNumerator, getDenominator, and display that return the values of the numerator, denominator of the fraction and display fraction. The class fraction should be able to overload operators which includes, +,-,*,/,+=,-=,*=,/=,<<,>>,==,!=,<,>,>=,<=,[],++,--,(),&&,||,&,->, ->* The header file and implementation files should be separate files.

Please also write down the test code to drive your class implementation. Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.

Q 2: Operator Overloading for String Class - Your goal is to overload the operators for String class that you have implemented in the last assignment. You will need to write three files (string.h, string.cpp and stringMain.cpp). Your implemented class must fully provide the definitions of following class (interface) functions. Please also write down the test code to drive your class implementation. Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.

```
#include<iostream>
using namespace std;

class String {
// think about the private data members...
public:
// provide definitions of following functions...
        String(); // default constructor
        String(char *str); // initializes the string with constant cstring
        String(const String &); // copy constructor to initialize the string
         ↪ fromexisting string
        String(int x); // initializes a string of pre-defined size
        char &operator[](int i); // returns the character at index [x]
        char &operator[](int i)const; // returns the character at index [x]

        String& operator+(const String &str); // append a String at the end of string
        String& operator+(const char &str); // append a char at the end of string
        String& operator+(char *&str); // append a String at the end of string

        String& operator-(const String &substr); //removes the substr from the string
        String& operator-(const string &substr); //removes the substr from the string


        bool operator!(); // returns true if string is empty..

        String& operator=(const String&); // Copy one string to another ...
        String& operator=(char*); // Copy one string to another ...
        String& operator=(const string&); // Copy one string to another ...


        bool operator==(const String&)const; //returns true if two strings are equal
        bool operator==(const string&)const; //returns true if two strings are equal
        bool operator==(char *)const; //returns true if two strings are equal


        int& operator()(char); // returns the index of character being searched.
        int& operator()(const String&); // returns the index of character being
         ↪ searched.
        int& operator()(const string&); // returns the index of character being
         ↪ searched.
        int& operator()(char *); // returns the index of character being searched.

        String operator*(int a); //multiples the string by i times and return the
         ↪ string. Remember the Python functionality for *
        int length(); // returns the length of string
        ~String(); // destructor...
};
ostream& operator<<(ostream& input, const String&); //Outputs the string
istream& operator>>(istream& ouput,  String&); //Inputs the string
```

Q 3: Operator Overloading for Matrix Class - Your goal is to overload the operators for Matrix class that you have implemented in the last assignment. You will need to write three files (matrix.h, matrix.cpp and matrixMain.cpp). Your implemented class must fully provide the definitions of following class (interface) functions. Please also write down the test code to drive your class implementation. Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.

```cpp
#include<iostream>
using namespace std;

class Matrix {
// think about the private data members...
// the matrix should store real numbers
public:
//include all the necessary checks before performing the operations in the functions
        Matrix(); // a default constructor
        Matrix(int, int); // a parametrized constructor
        Matrix(const Matrix &); // copy constructor
        float &operator()(int &i, int &j); //set value at (i,j)
        float &operator()(int &i, int &j)const; //set value at (i,j)
        Matrix& operator=(const Matrix &); //assigns (copies) a Matrix. Returns the same
        bool operator==(const Matrix &); //Compares two matrices
        Matrix operator+(const Matrix &); //adds two Matrices and returns the result
        Matrix operator-(const Matrix &); //subtracts two Matrices and returns the result
        Matrix operator*(const Matrix &); //multiplies two Matrices
        Matrix& operator++(int); //add one to every element
        void operator+=(const Matrix&); //adds two Matrices
        void operator-=(const Matrix&); //subtracts two Matrices
        ~Matrix();
};
ostream& operator<<(ostream& input, const Matrix&); //Outputs     The Matrix
istream& operator>>(istream& ouput, Matrix&); //Inputs     the Matrix
```

Q 4: Operator Overloading for Polynomial Class - Your goal is to overload the operators for a generic Polynomial class. A polynomial will be represented via its coefficients. Here is a degree of third degree polynomial 4x3+ 3x + 2; here will be four coefficients and the coefficient corresponding to 2nd power is zero. You will need to write three files (polynomial.h, polynomial.cpp and polynomialMain.cpp). Your implemented class must fully provide the definitions of following class (interface) functions. Please also write down the test code to drive your class implementation. Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.

```cpp
#include<iostream>
using namespace std;

class Polynomial {
// think about the private data members...
// the matrix should store real numbers
public:
//include all the necessary checks before performing the operations in the functions
        Polynomial(); // a default constructor
        Polynomial(int); // a parametrized constructor, received the highest degree of polynomial
        Polynomial(const Polynomial &); // copy constructor
        Polynomial& operator=(const Polynomial &); //assigns (copies) a Polynomial. Returns the same
        bool operator==(const Polynomial &); //Compare and return true if equal
        Polynomial operator+(const Polynomial &); //adds two Polynomial and returns the result
        Polynomial operator-(const Polynomial &); //subtracts two Polynomial and returns the result
        void operator+=(const Polynomial&); //adds two polynomials
        void operator-=(const Polynomial&); //subtracts two Polynomials
        ~Polynomial();
};
ostream& operator<<(ostream& input, const Polynomial&); //Outputs the Polynomial
istream& operator>>(istream& ouput, Polynomial&); //Inputs the Polynomial
```

Q 5: Operator Overloading for Array Class - Your goal is to overload the operators for Array class. You will need to write three files (array.h, array.cpp and arrayMain.cpp). Your implemented class must fully provide the definitions of following class (interface) functions. Please also write down the test code to drive your class implementation. Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.

```
#include<iostream>
using namespace std;

class Array {
// think about the private data members...
public:
// provide definitions of following functions...
        Array(); // a default constructor
        Array(int size); // a parametrized constructor initializing an Array of
         ↪ predefined size
        Array(int *arr, int size); // initializes the Array with an existing Array
        Array(const Array &); // copy constructor
        int& operator[](int i); // returns the integer at index [i] after checking
         ↪ the out of range error
        int& operator[](int i)const;
        const Array & operator=(const Array&); //copy the array
        Array operator+(const Array&); //adds two Array
        Array operator-(const Array&); //subtracts two Array
        Array operator++(); //adds one to each element of Array
        Array operator++(int); //adds one to each element of Array
        Array& operator--(int); //subtracts one from each element of array
        bool operator==(const Array&)const; //returns true if two arrays are same
        bool operator!(); // returns true if the Array is empty
        void operator+=(const Array&); //adds two Array
        void operator-=(const Array&); //subtracts two Array
        int operator()(int idx, int val); // erases the value val at idx. Returns 1
         ↪ for a successful deletion and -1 if idx does not exists or is invalid.
         ↪ Shift the elements after idx to the left.
        ~Array(); // destructor...
};
ostream& operator<<(ostream& input, const Array&); //Inputs the Array
istream& operator>>(istream& ouput, Array&); //Outputs the Array
```