Lab # 10

Operator overloading as Non member

TASK 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.

TASK 2

Operator Overloading for String Class - Your goal is to overload the operators for String class. 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.

```cpp
#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
```