CECS 282 – Assignment 3

Complex Calculator

A complex number can be represented in the form a + bi where a and b are real numbers. In this assignment, only the form a + bi, a – bi, and bi, where a and b are positive real numbers, are accepted. For instance, 6 + 5i, 6 – 5i, and 5i are acceptable forms. Let's refer to these as "standard complex numbers". In addition, it is possible to have complex number such as 2/3 + 1/2i, which is referred as "non-standard complex number" and represented by [a + bi]/r format where a and b are non-negative integers and r is a non-zero integer. For instance 2/3 + 1/2i is formatted as [4 + 3i]/6.

Complex arithmetic consists of addition ('+'), subtraction ('-'), multiplication ('*'), division ('/'), and conjugate ('%'). For example:
(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)
(a + bi)% = a – bi

Precedence of operators: '%' has the highest precedence. '+' and '-' have the next precedence. '*' and '/' have the lowest precedence. Operator with the same precedence are evaluated in a left-to-right order.

Your task is to design and implement several classes.

```
template <class T>
class DoublyLinkedList
{
    protected:
        Node<T>* head; //pointer to the first node of a doubly linked list
        Node<T>* tail; //pointer to the last node of a doubly linked list
        int size; //size of a doubly linked list
    public:
        DoublyLinkedList();
        void addFront(T d); //add a node at the beginning of a doubly linked list
        void popFront(); //remove a node at the beginning of a doubly linked list
```

```
class Complex
{
    public:
        Complex();
        Complex(int r, int i, int d);
        int re; //real part of a complex number
        int im; //imaginary part of a complex number
        int dem; //denominator part of a complex number
        string toString() const; //Format [re + imi]/dem

};
```

```
        void addBack(T d); // add a
node at the end of a doubly linked
list
        void popBack(); // remove a
node at the end of a doubly linked
list
        void addNode(T d, Node<T>*
iter); //add a node in general
        void deleteNode(T d, Node<T>*
iter); //delete a node in general
        Node<T>* findNode(T d);
//return a pointer to a node in a
doubly linked list and return NULL
otherwise
        int getSize() const; //return
size of a doubly linked list
        bool isEmpty() const; //return
true if empty and false otherwise
        void displayList()
const;//display a doubly linked list
};
```

```
//Overloading operator for easy
arithmetic
Complex operator+(Complex a, Complex
b);
Complex operator-(Complex a, Complex
b);
Complex operator*(Complex a, Complex
b);
Complex operator/(Complex a, Complex
b);
ostream& operator<< (ostream& stream,
Complex a);

//Find gcd and lcm to reduce fraction
and add fraction
int gcd(int a, int b);
int lcm(int a, int b);
```

```
template <class T>
class StackDoublyLinkedList : public
DoublyLinkedList<T>
{
    public:
        StackDoublyLinkedList();
        void displayStack() const;
//Display a stack
        T getTop() const; //return
element at the top of a stack
};
```

```
template <class T>
struct Node
{
    T data; //data of a node
    Node* prev; //pointer to previous
node
    Node* next; //pointer to next node
};
```

Input:
All inputs for this assignment consist of algebraic expression of complex numbers and terminate with an '=' sign. Operands of these expression can be either a positive number or a standard complex number. You will create a file named *expression.txt* to store all algebraic expression. Also, suppose there are no spaces between operands and operators.

Sample input:

```
33+2i*4+i=
22+i*22-i =
(2+3i)/(7+i)%=
(5+3i)/(1+3i)%=
(1+i)%/3+i*2-i+3=
((1+i)%/3+i*2-i+3)+(3i+1*(2i*2-i%/3))=
9+(1-i)*(1+i)=
```

Output:
Your program will execute all expressions in the *expression.txt* file and produce a *result.txt* file in which contains all calculated answer. The format of each answer is the same as non-standard complex number.

Sample output:

```
[130+41i]/1
[485+0i]/1
[11+23i]/50
[2-9i]/-5
[3-11i]/5
[61+43i]/-15
[11+9i]/1
```

NOTE
- Please include the following block at the beginning of your program

/*
Name:
Class: CECS 282
Instructor: Minhthong Nguyen
Purpose of the program:
Last updated:
*/

- Comment your code.
- Follow standard style for coding (refer to java docs).

<u>Deliverables:</u>
Turn-in all files (header, cpp, and txt) to Dropbox and bring a physical copy of all files (header, cpp, and txt) when you demo your program.