CSE 108 (January 2025)

Offline 01

Implement the following classes.

Fraction:

Implement a class Fraction that models a fraction and supports basic arithmetic operations on fractions. You have to reduce the fraction to its simplest form (e.g., 24/4 => 6/1). You can show it as 6/1 or 6; both are acceptable.

Data members:

1. int numerator - Represents the numerator of the fraction.
2. int denominator - Represents the denominator of the fraction.

Class Functionality:

1. Constructors
    a. Default Constructor: Initializes the fraction to 0/1.
    b. Constructor with Single Integer: The integer is used as the numerator. The denominator is 1 here.
    c. Constructor with Two Integers
2. Destructor
    a. Handles cleanup when the object is destroyed if necessary.
3. Member Functions
    a. Addition (add):
        i. Fraction add(Fraction &f): Adds another Fraction object to the current fraction.
        ii. Fraction add(int n): Adds an integer n to the current fraction.
    b. Subtraction (sub):
        i. Fraction sub(Fraction &f): Subtracts another Fraction object from the current fraction.
        ii. Fraction sub(int n): Subtracts an integer n from the current fraction.
    c. Multiplication (mul):
        i. Fraction mul(Fraction &f): Multiplies the current fraction with another Fraction object.
        ii. Fraction mul(int n): Multiplies the current fraction with an integer n.
    d. Division (div):
        i. Fraction div(Fraction &f): Divides the current fraction by another Fraction object.
        ii. Fraction div(int n): Divides the current fraction by an integer n.
    e. print(): Prints the Fraction object.

FractionCollection:

Implement a class FractionCollection that manages a collection of Fraction objects.

Data members:

1. **Fraction* fractions** - A dynamic array to hold the Fraction objects.
2. **int maxlength** - The maximum number of elements that the array can currently hold.
3. **int length** - Current number of elements.

Class Functionality:

1. Constructors
    a. **Default Constructor**: Initialize an empty FractionCollection with an initial capacity (10).
    b. **Constructor with Maximum Size**: Initialize a FractionCollection with a user-defined maximum size for the array.
2. Destructor
    a. Properly free the dynamically allocated memory when the FractionCollection object is destroyed.
3. Member Functions
    a. Insert:
        - **void insert(Fraction f):** Adds a Fraction object to the end of the collection.
        - **void insert(int pos, Fraction f):** Adds a Fraction object at a specific position in the collection.
    b. Remove:
        - **void remove():** Removes the last Fraction object from the collection.
        - **void remove(Fraction f):** Removes the Fraction object from the collection.
        - **void remove(int pos):** Removes the Fraction object from the specified position.
    c. **Fraction getmax():** Returns the Fraction object with the maximum value.
    d. **Fraction getmin():** Returns the Fraction object with the minimum value.
    e. **Fraction add(int start, int end)**: Adds all the Fraction objects between the specified positions.
    f. **Fraction mul(int start, int end)**: Multiplies all the Fraction objects between the specified positions.
    g. **Fraction sub(int pos1, int pos2)**: Subtracts the fraction at pos2 from the fraction at pos1.
    h. **Fraction div(int pos1, int pos2)**: Divides the fraction at pos1 by the fraction at pos2.
    i. **print():** Displays information about all the stored Fraction objects.

* You have to use the member functions from the previous classes to implement the functionalities.

Sample Main Function

```
int main(){
    //create Fraction with numerator, denominator
    Fraction a(5,2),b(7,2),c(9,2),d(28,5);
    cout<<"Fraction"<<endl;
    cout<<"------------------------------"<<endl;
    cout<<"A: ";
    a.print();
    cout<<"B: ";
    b.print();
    cout<<endl;

    cout<<"Add(a,b): ";
    a.add(b).print();
    cout<<"Add(a,2): ";
    a.add(2).print();

    cout<<"Sub(a,b) ";
    a.sub(b).print();
    cout<<"Sub(a,2) ";
    a.sub(2).print();

    cout<<"Mul(a,b): ";
    a.mul(b).print();
    cout<<"Mul(a,2): ";
    a.mul(2).print();

    cout<<"Div(a,b): ";
    a.div(b).print();
    cout<<"Div(a,2): ";
    a.div(2).print();
    cout<<"Div(a,0): ";
    a.div(0).print();


    //Collection of Fractions
    Fraction e,f(5),g(10);
    FractionCollection fc(10);
    fc.insert(a);
    fc.insert(b);
    fc.insert(c);
    fc.print();

    cout<<"Sub(Pos0, Pos1): ";
    fc.sub(0,1).print(); //subtracts the fraction at pos1 from fraction at pos0
```

```
    cout<<"Div(Pos0, Pos1): ";
    fc.div(0,1).print(); //divides the fraction at pos0 by the fraction at pos1

    fc.remove(1);  //removed 'b'
    fc.print();

    fc.remove(a);
    fc.print();

    fc.insert(d);
    fc.insert(0,e);  //insert at pos0
    fc.insert(f);
    fc.insert(g);
    fc.print();

    fc.remove(); //removed the last fraction
    fc.print();  //notice the output

    return 0;
}
```

Expected Output

```
Fraction
--------------------------------
A: 5/2
B: 7/2

Add(a,b): 6/1
Add(a,2): 9/2
Sub(a,b) -1/1
Sub(a,2) 1/2
Mul(a,b): 35/4
Mul(a,2): 5/1
Div(a,b): 5/7
Div(a,2): 5/4
Div(a,0): Can not divide by 0
5/2

Fractions
--------------------------------
Fraction 0: 5/2
Fraction 1: 7/2
Fraction 2: 9/2
Max: 9/2
Min: 5/2
Summation: 21/2
Multiplication: 315/8
Sub(Pos0, Pos1): -1/1
Div(Pos0, Pos1): 5/7
```

```
Fractions
--------------------------------
Fraction 0: 5/2
Fraction 1: 9/2
Max: 9/2
Min: 5/2
Summation: 7/1
Multiplication: 45/4

Fractions
--------------------------------
Fraction 0: 9/2
Max: 9/2
Min: 9/2
Summation: 9/2
Multiplication: 9/2

Fractions
--------------------------------
Fraction 0: 0/1
Fraction 1: 9/2
Fraction 2: 28/5
Fraction 3: 5/1
Fraction 4: 10/1
Max: 10/1
Min: 0/1
Summation: 251/10
Multiplication: 0/1

Fractions
--------------------------------
Fraction 0: 0/1
Fraction 1: 9/2
Fraction 2: 28/5
Fraction 3: 5/1
Max: 28/5
Min: 0/1
Summation: 151/10
Multiplication: 0/1
```

You can copy the main() function shown above for your convenience.

Mark distribution

Correct Implementation of the Fraction Class	40%
Correct Implementation of the FractionCollection Class	50%
Properly allocate and free memory	10%

Submission guidelines:

1. Create a folder named after your student ID.

2. Write your code in a single file named your_ID.cpp.

3. Move the .cpp file into the folder from step1.

4. Zip the folder and name it after your student ID. Submit this zip on Moodle.

For example,

2305xyz/

└──2305xyz.cpp

Then submit 2305xyz.zip

Important Note:

Keep the member variables of the classes 'private', use appropriate getters and setters to access the variables. You can use additional variables, methods as needed.

Suggestion: If you find it difficult to write multiple classes at once, do it step by step. Start by implementing the **`Fraction`** class and testing it with the corresponding part of **`main()`** . Once it's working, proceed with the **`FractionCollection`** class to manage the collection of lines.

Deadline: April 28 (8:00 AM)

Please do not copy from any sources (friends, internet, AI tools like ChatGPT, etc.). **Doing so may result in severe penalties.**