Ronish Shrestha
19707
CS360LAB_Assignment4

Q1)

a) Correction: ~Time();

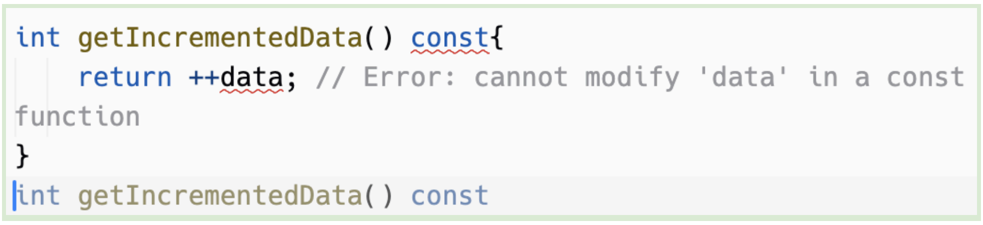
Destructors in C++ cannot take parameters and should not have a return type. They are defined using only the class name prefixed with a tilde (~). The corrected declaration simply removes the parameter and return type:

b) Correction: Employee(string, string);

Constructors in C++ do not have a return type, not even void. They are named after the class and can have parameters but should not specify a return type. Constructors are used to initialize objects and cannot return values.

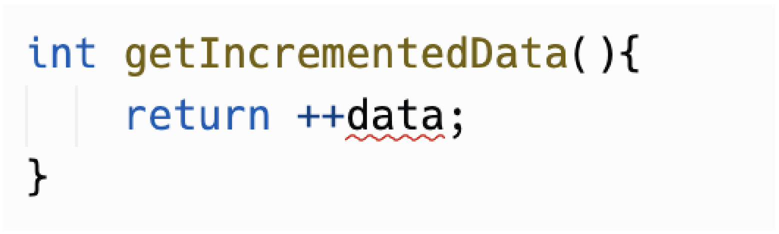
c)

Error:



```
int getIncrementedData() const{
    return ++data; // Error: cannot modify 'data' in a const
function
}
int getIncrementedData() const
```

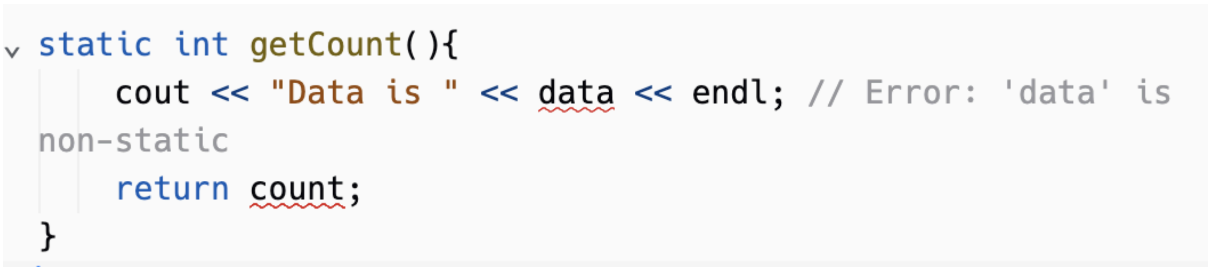
Correction:



```
int getIncrementedData(){
    return ++data;
}
```

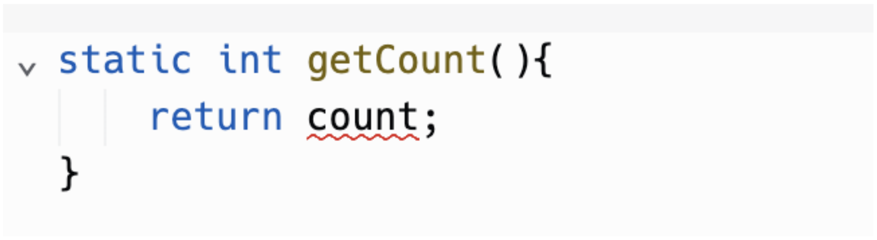
Attempting to modify a member variable in a const member function violates the promise that the function will not alter the object's state.

Error:



```
static int getCount(){
    cout << "Data is " << data << endl; // Error: 'data' is non-static
    return count;
}
```

Correction:



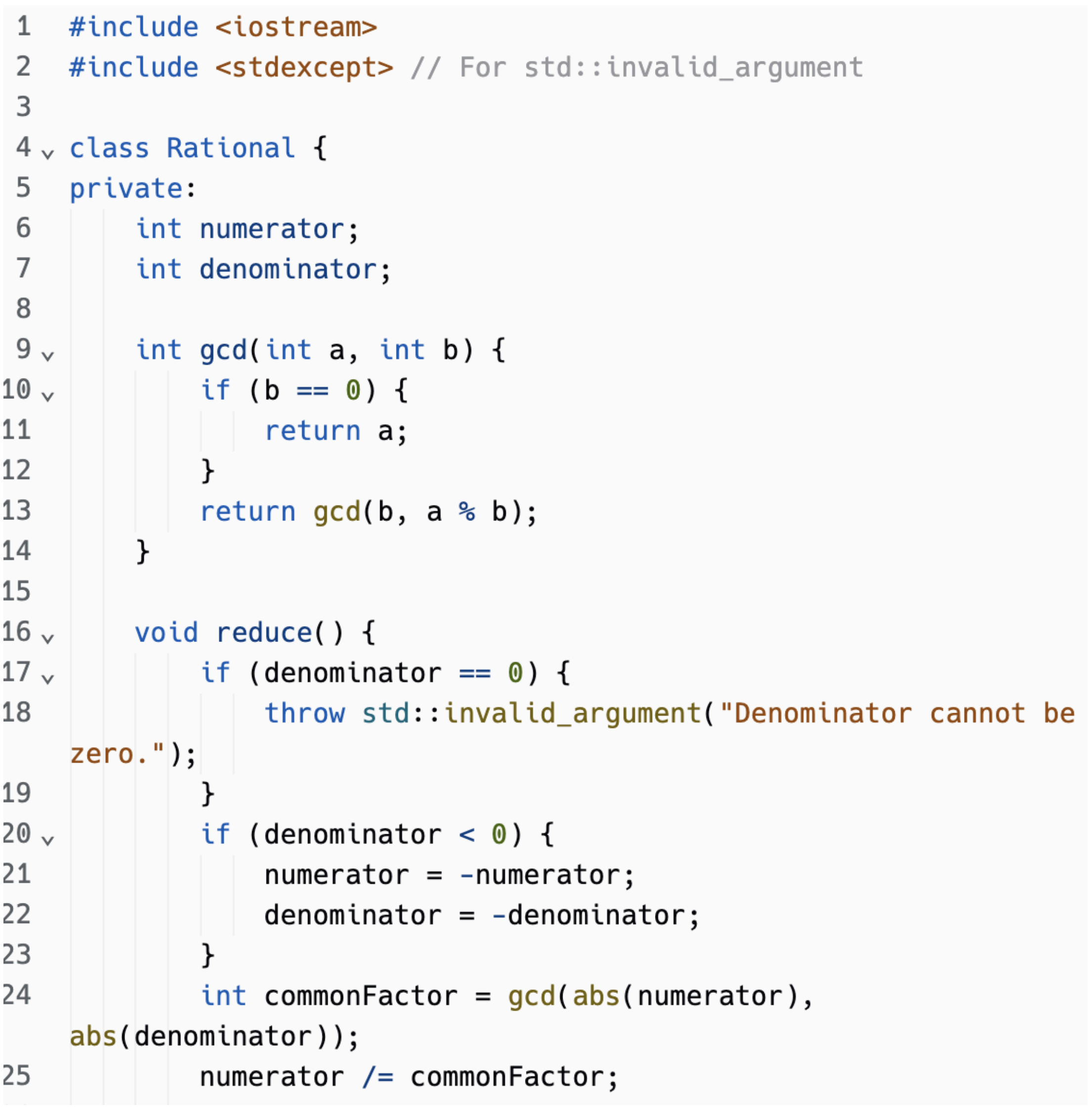
```
static int getCount(){
    return count;
}
```

Static member functions can only access static data members and other static functions within the class because they are not associated with any particular instance of the class.

Correct snippet:

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

class Example {
public:
    Example(int y = 10) : data(y) {}

    // If it's logical for the object's state to be mutable, use mutable keyword or remove const
    int getIncrementedData() {
        return ++data; // This function now correctly indicates it modifies the object.
    }

    // This function cannot access non-static member variables like `data` directly.
    static int getCount() {
        // Cannot print "Data is " since 'data' is non-static. This function now only deals with static members.
        cout << "Count is " << count << endl;
        return count;
    }

private:
    int data; // Instance-specific data
    static int count; // Shared among all instances
};
```

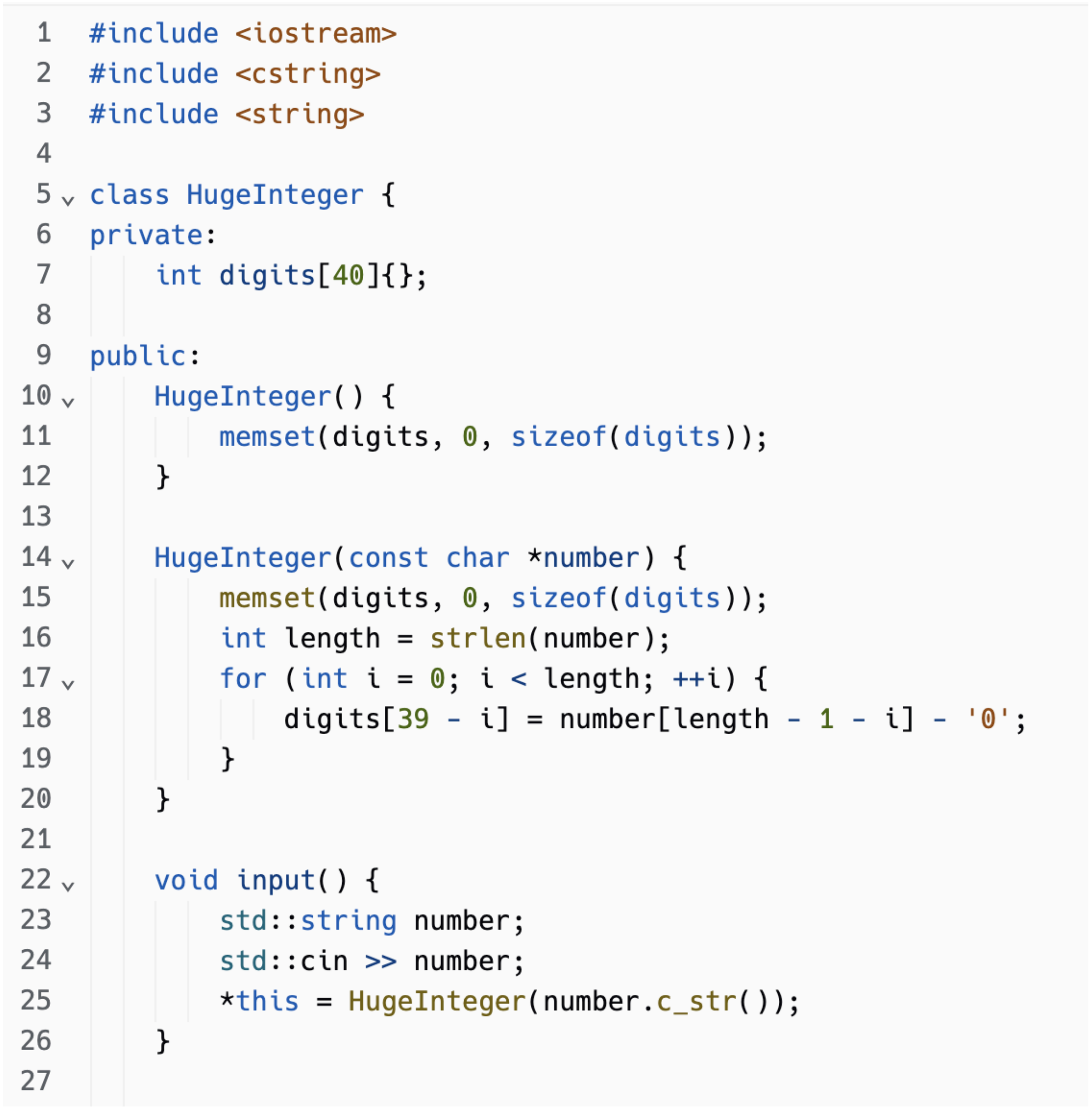
Q2)



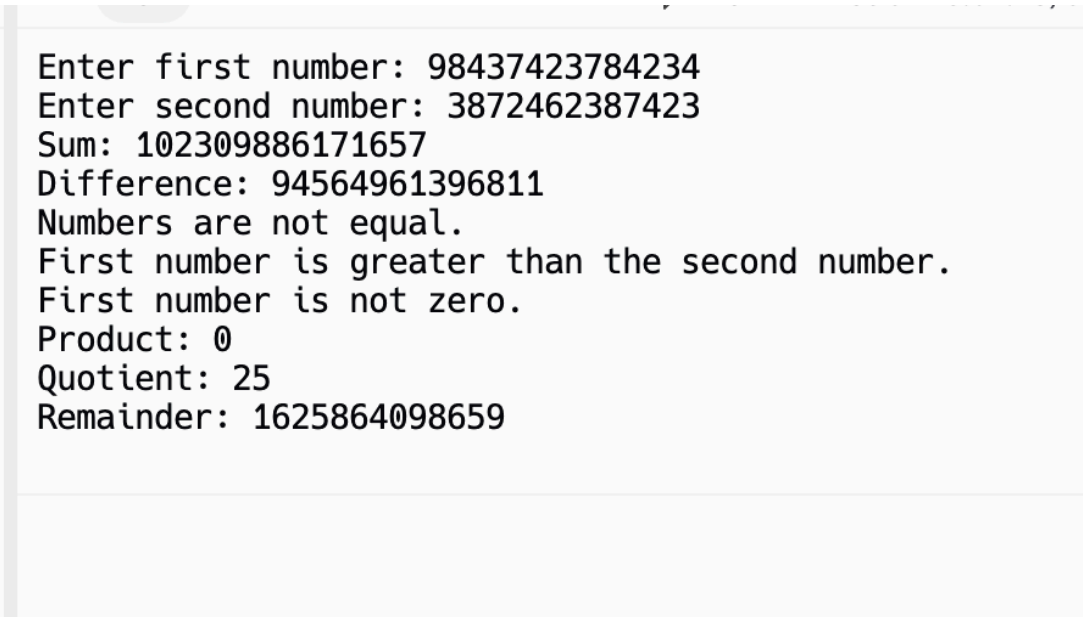
```cpp
#include <iostream>
#include <stdexcept> // For std::invalid_argument

class Rational {
private:
    int numerator;
    int denominator;

    int gcd(int a, int b) {
        if (b == 0) {
            return a;
        }
        return gcd(b, a % b);
    }

    void reduce() {
        if (denominator == 0) {
            throw std::invalid_argument("Denominator cannot be zero.");
        }
        if (denominator < 0) {
            numerator = -numerator;
            denominator = -denominator;
        }
        int commonFactor = gcd(abs(numerator), abs(denominator));
        numerator /= commonFactor;
```

```
Enter numerator and denominator for the first rational
number (r1): 1
2
Enter numerator and denominator for the second rational
 number (r2): 1
4
Sum: 3/4
Difference: 1/4
Product: 1/8
Quotient: 2/1
R1 Float: 0.5
R2 Float: 0.25
```

Q3)

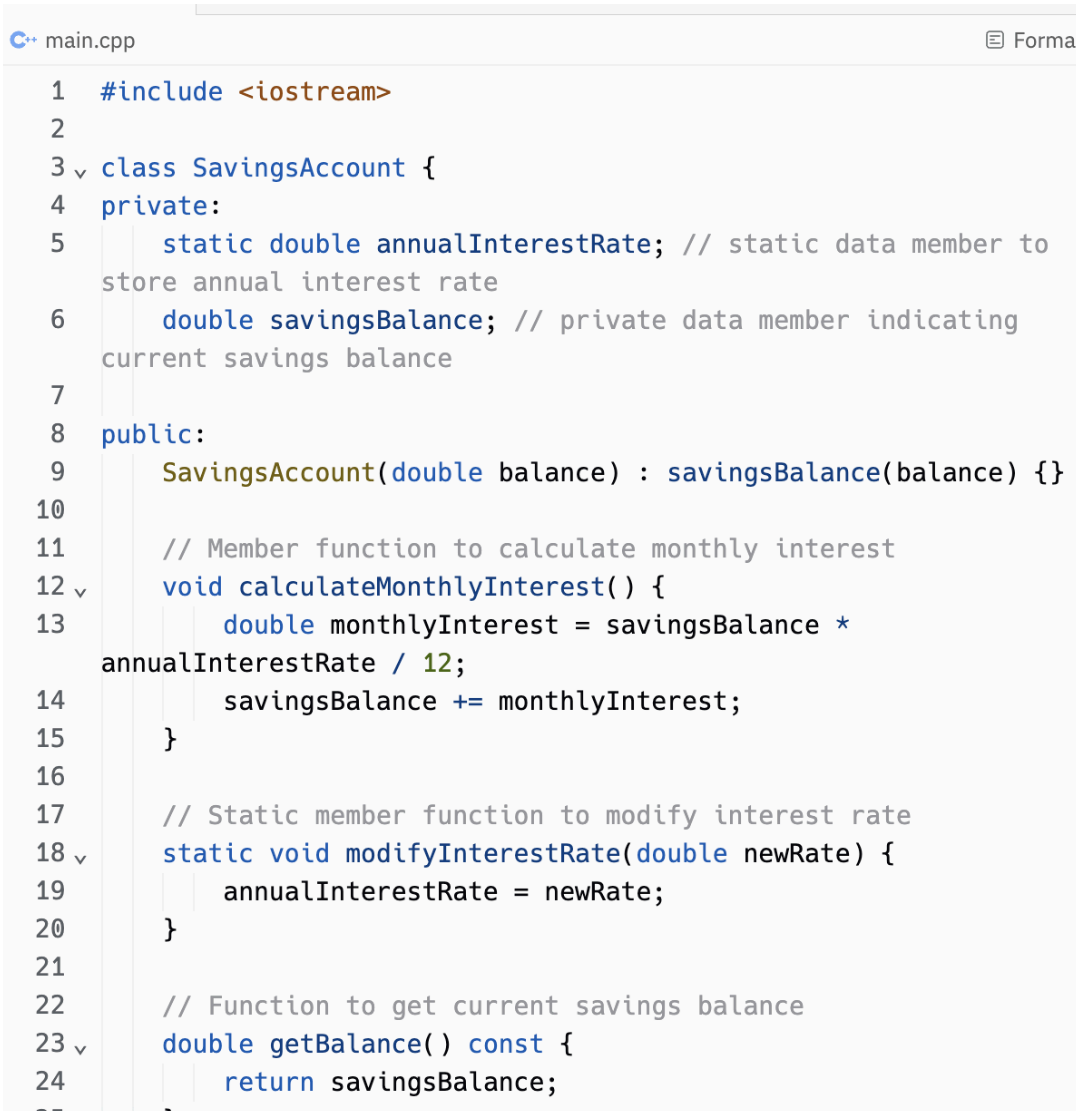


```cpp
#include <iostream>
#include <cstring>
#include <string>

class HugeInteger {
private:
    int digits[40]{};

public:
    HugeInteger() {
        memset(digits, 0, sizeof(digits));
    }

    HugeInteger(const char *number) {
        memset(digits, 0, sizeof(digits));
        int length = strlen(number);
        for (int i = 0; i < length; ++i) {
            digits[39 - i] = number[length - 1 - i] - '0';
        }
    }

    void input() {
        std::string number;
        std::cin >> number;
        *this = HugeInteger(number.c_str());
    }

```

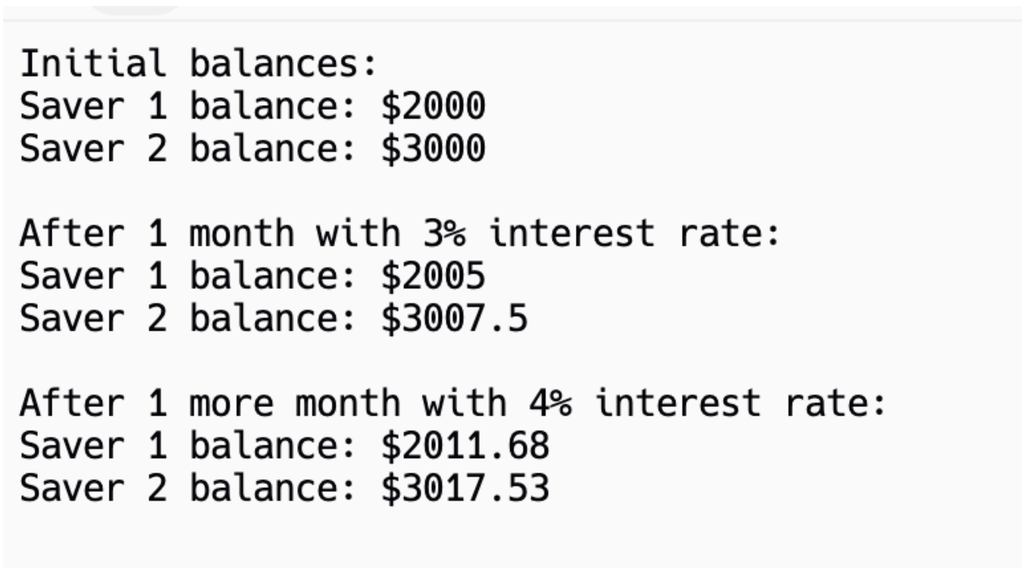


```
Enter first number: 98437423784234
Enter second number: 3872462387423
Sum: 102309886171657
Difference: 94564961396811
Numbers are not equal.
First number is greater than the second number.
First number is not zero.
Product: 0
Quotient: 25
Remainder: 1625864098659
```

Q4)



```cpp
#include <iostream>

class SavingsAccount {
private:
    static double annualInterestRate; // static data member to store annual interest rate
    double savingsBalance; // private data member indicating current savings balance

public:
    SavingsAccount(double balance) : savingsBalance(balance) {}

    // Member function to calculate monthly interest
    void calculateMonthlyInterest() {
        double monthlyInterest = savingsBalance * annualInterestRate / 12;
        savingsBalance += monthlyInterest;
    }

    // Static member function to modify interest rate
    static void modifyInterestRate(double newRate) {
        annualInterestRate = newRate;
    }

    // Function to get current savings balance
    double getBalance() const {
        return savingsBalance;
```



```
Initial balances:
Saver 1 balance: $2000
Saver 2 balance: $3000

After 1 month with 3% interest rate:
Saver 1 balance: $2005
Saver 2 balance: $3007.5

After 1 more month with 4% interest rate:
Saver 1 balance: $2011.68
Saver 2 balance: $3017.53
```