Programming Assignment #1: C++ Huge Integer

CS 3100 – Programming Languages -- Spring 2023

Due: Thursday, February 16th at midnight

Goals for this assignment

- Practice using C++ classes.
- Gain experience with operator overloading.

Academic Honesty

Please read carefully the section titled “Academic Integrity” in the syllabus. If you have any questions, contact your instructor before you begin this assignment. This is an individual assignment

Requirements

Write a C++ program that implements basic arithmetic with large integers. The core of your program should be a class called **`HugeInteger`**. Your HugeInteger class must include the ability to store an arbitrary number of digits. You may choose to use the C++ STL class **`std::vector`** to store the digits of your number.

If your class allocates any dynamic memory on the heap, your class must include a destructor.

Implement your class using a header file (**`HugeInteger.h`**) and an implementation file (**`HugeInteger.cpp`**). Implement the following methods in your HugeInteger class:

- **`HugeInteger()`** - Constructor, sets the value to 0.
- **`HugeInteger( const HugeInteger & other )`** - Copy constructor. Makes a deep copy of other.
- **`HugeInteger( const string & s )`** - Initializes this HugeInteger to have the value of the string of digits in s. If s contains characters that are not digits, then you should ignore them. In other words, if the string is “123abc456”, then the object will be initialized with a value of 123456.
- **`bool operator==( const HugeInteger & rightSide )`** - Overloads the == operator to return whether this object has the same value as rightSide.
- **`const HugeInteger operator+( const HugeInteger & rightSide )`** - Overload the binary + operator to return the sum of this and rightSide.
- **`bool operator>( const HugeInteger & rightSide )`** - Overloads the > operator to return whether this is greater than rightSide.
- **`bool isZero() const`** - Returns whether or not this object has a value of 0. Remember that 0 == 00 == 000 ...
- **`string toString() const`** - Returns a string representation of this number.

Write a main method in a separate file that tests all of the above functions. Your program will be tested using my own tests, so make sure that you are confident in the functionality of each of your methods. Your program should include 3 files: **`HugeInteger.cc`**, **`HugeInteger.h`**, and **`main.cc`**.

Hints

For the **`toString`** method, you might find it useful to use the **`stringsstream`** class. This makes it easy to build C++ **`string`** objects using the **`<<`** operator. For example:

```cpp
#include <sstream>
using std::stringstream;
#include <string>
using std::string;

...
int age = 340;
string fName = "Zaphod", lName = "Beeblebrox";
char mi = 'J';

stringstream stream;
stream << "Hello, my name is " << fName << " " << mi << " "
       << lName << " and my age is " << age;
string result = stream.str();  // Convert stream to a C++ string object
```

To access individual elements of strings and vectors, you can use the array index operator ([]). For example:

```cpp
#include <string>
#include <vector>
#include <iostream>
using std::cout;
using std::endl;

...

std::string name = "Arthur Dent";
cout << name[2] << endl;   // Output: t
cout << name[0] << endl;   // Output: A

std::vector<int> myVec;
myVec.push_back(42);
myVec.push_back(24);

cout << myVec[0] << endl;  // Output: 42
cout << myVec[1] << endl;  // Output: 24
```

Submission Instructions

- Add a comment header to the top of your main source code file including your name, date, sources used, known bugs, and any special instructions.
- Submit your code (3 files) using Canvas.