COP3330 Programming Assignment 3

The bigint class

Objective

1. Practice building class with operator overloading.

Overview

The native *int* type in C++ cannot store large integers of more than 10 decimal digits. In this project, you will implement a bigint class that behaves like the int type, but can handle unsigned integers of up to 45 digits.

Details

1. You are to build a class named *bigint* that can deal with up to 45 decimal digits of unsigned integers. The *bigint* class is specified as follows.
2. The *bigint* class should have a five-element array (*int v[5];*) as its private date member. Each element should store 9 decimal digits. For example to store integer 1,300,000,000,000,900,000,000,100,000,000. You should have v[0] = 100000000, v[1] = 900000000, v[2] =0, v[3]= 1300, and v[4] = 0.
3. The bigint class should have the following constructors:

 bigint(); // default constructor, set the value to be 0

 bigint(int x0); // set the value to be x0

 bigint(int x0, int x1); // set the value to be x1\*109+x0

 bigint(int x0, int x1, int x2); // set the value to be x2\*1018+x1\*109+x0

 bigint(int x0, int x1, int x2, int x3);

 bigint(int x0, int x1, int x2, int x3, int x4);

1. To make bigint behave like int, you must overload the following operators
	1. The extraction operator >> for reading a bigint number from an input stream. The input number can have up to 45 digits. Following are two legitimate bigint numbers that can appear as input:

1300000000000900000000100000000

9999999999999999999999999999999999999999999

* 1. The insertion operation << for output a bigint number to an output stream. The number must be output like a regular int number with the first digit being a non zero.
	2. 2 arithmetic operators + and – (\*, and / are omitted for this type to simplify the program).
	3. 6 comparison operators <, >, <=, >=, ==, and !=.
1. Once the class is implemented, you can test your implementation using the proj3\_driver.cpp program. You should also write other driver programs to do a thorough test of your bigint implementation. Put your bigint class declaration in proj3\_bigint.h and your bigint class implementation in proj3\_bigint.cpp. You should be able to compile the program by the command ‘g++ proj3\_driver.cpp proj3\_bigint.cpp’. Redirect proj3\_test.txt as the standard input, the output of the program should be identical to proj3\_test\_out.txt.

Submission

The due time for this assignment is June 5 (Wendesday), 2013. 11:59pm.

Tar all of your files for this assignment, which should include at least three files proj3\_bigint.h, proj3\_bigin.cpp, and bug\_fixing\_log.txt, name the tar file yourlastname\_firstinitial\_proj3.tar and submit the tar file in blackboard. The bug fixing log file must follow the template given in project 1.

Grading policy

The program must work on linprog. O point for programs with any g++ compiler error on linprog. You will need to track compiling errors that **you fixed by yourselves** in a log for fixed compiler bugs that needs to have at least two entries for 5 points/each entry (10 points max) that are different from those in the bug fixing log in projects 1 and 2.

* Program with no compiler error (20 points)
* Log for fixed compiler bugs (10 points)
* The default constructor and the overloaded << operator (20 points)
* Other four constructors (3 points each, 12 points total)
* Six comparison operators (2 points each, 12 points total)
* + and – operators (8 points each, 16 point total)
* The >> operator (10 points)

Hints

1. Start the project as soon as possible.
2. You should first create proj3\_bigint.h and have an empty implementation in proj3\_bigint.cpp for each member and friend function with the body of the function containing one line

 *cout << “XXXX function has not been implemented.\n” ;*

1. You should then compile the proj3\_bigint.cpp and proj3\_driver.cpp and run the code. From here, you can implement and test one function after another until the whole class is implemented.
2. You should then implement the default construct, and write a simple operator<<() function that simply prints out the values of v[0], v[1], v[2], v[3], v[4].
3. After that you can implement the functions in the following order: the << operator, other constructors, the comparison operators, the arithmetic operators, and the >> operator. Remember to compile and run the program at least once everytime you add one function.
4. To implement the << operator, you will need to output potentially 5 integers. You must use setw(9) and setfill(‘0’) to output each value besides the most significant non-zero integer. You must have ‘#include <iomanip>’ in the file in order to use these functions. The following line is an example to use these functions.

s << setfile(‘0’) << setw(9) << vv.v[j];

1. To implement the >> operator, you can first read the string of digits into a character array and then manually convert it into a bigint number.
2. To implement the + and – operators, you must pay special attention to the carry bit in the operations.
3. To see the prototype of the overloaded operators (+, ==, >>, <<), check the examples in lecture 6.