Santa Clara University
School of Engineering

CSEN 79-2: Mid-Term #2, Coding

Assignments

Finish the "`BigNum`" class, as provided in `bignum.tar`.

Specifically:

- implement the "rule of 4" member functions (destructor, copy constructor, assignment operator, move operator), another constructor that accept a "`long`" argument, and the "+" operator.
- Design an automated test to cover the code execution paths and the boundary conditions. Submit the source code, including the test scripts, if you chose to write them. Note a sample test data file with many big numbers is part of the tar file.

  The test script should be written in Python or Unix shell. It should "feed" the executable with data from an external file (not in the script file). It should have the mechanism to feed multiple test data files without altering the script. For each such test data file, it should output "pass" or "fail".
- Submit all source files in a tar or zip package.

Additionally:

- Consider the alternatives of "`store_t`" type (it is "unsigned char" in provided code). Prepare to answer in the in-person part of mid-term. (It will be close-book, close-note.)

Background

An integral value can be mathematically represented by a polynomial.

$v = sign * \sum_{i=0}^{n} x_i B^i$, where sign = ± 1, $0 \leq x_i < B$ and B is the "base" which is an integer usually greater than or equal to 2.

For common modern C++ environments, n is the number of 7, 15, 31, or 63 for integral types `char`, `short`, `int`, or `long`. The most significant bit represent sign where zero is positive and 1 negative. (The lengths for `int` and `long` are common, not guaranteed.)

This assignment asks you to complete the design for a class, called **`BigNum`**, that effectively removes the length limitation for C++ integral types. The provided code designed a container with an internal storage that is extensible and utilize the polynomial mathematics for arithmetic operations.

Hints

The provided code can convert a string of decimal digits, effectively unlimited in length, to a BigNum object. It can also output the internal storage in the format suitable for verification. These construction and formatted output algorithms are usefule to the addition operator. To add two polynomials, one only need to loop through them and add the coefficients of the same order. Since we have a slot for each order, the code is relatively straight-forward. The tricky part is when the sum of the two coefficients exceeds the base. In that case, one must "carry" to the higher order and reduce the current one appropriately.

Santa Clara University
School of Engineering

CSEN 79-2: Mid-Term #2, Coding

The provided code put `operator+` "inside" as a member function. You can change it to "outside" as a friend function.

The constructor uses "`checkCapacity`" which may throw exception. Your code should accommodate this. One way is to change the storage of digits from pointer to vector. If you choose to do so, make sure you check the provided code to be still working (and fix them if need to.)

Explanation for conversion from string

The code takes the incoming decimal string from left to right. After consuming the optional sign character, each digit effectively multiplies the integral value by 10, before adding the new character's decimal value.

As digits get a new value, it is important to avoid overflow. We do so by using a buffer that's twice as long as the digit. If we found the resulting value greater than the maximum of the storage unit, we "carry" the excess to the next higher order digit. This "compute and carry if necessary" operation is repeated or cascading up until we have done all the processing. The addition operation must do similarly.

The formatted output operator (<<) was crudely done. It outputs the digits in polynomial form meant for debugging. Consider changing it to output in decimal, hexadecimal, or octal notations.