Задача #2. Длинные целые числа

Напишите класс целого длинного числа со знаком, обладающего **как минимум** следующими операциями:

1. неявное конструирование от числа:
    a. целого,
    b. вещественного - конструирование должно отсекать дробную часть, но сохранять целые разряды (например lint(2.34e20) превратится в 234000000000000000000);
2. явное конструирование от строки;
3. копирование;
4. явные преобразования к числу;
5. возможность использования в условных выражениях (true, если не ноль);
6. функция получения строкового представления;
7. ввод/вывод в стандартные потоки (std::ostream/std::istream), ввод/вывод аналогичен обычным целым числам (при вводе могут присутствовать как унарный +, так и ведущие нули);
8. операции вида @= и @ (где @ - это +, -, *, /);
9. операции сравнения (все 6);
10. постфиксный/префиксный инкремент/декремент;
11. а также несколькими вспомогательными функциями, описанными ниже.

Ообратите внимание, для компиляции smoke-теста, предложенного в этом задании, Вам можем потребоваться дополнительные операции помимо указанных в списке выше.

Дополнительные требования:

1. Класс должен называться lint и находится в пространстве apa (arbitrary-precision arithmetic).
2. Ваше решение должно содержать ровно два файла: lint.h с объявлениями и lint.cpp с определениями. Никаких makefile'ов не требуется.
3. Деление длинного целого числа на ноль приводит к тому же эффекту, что и деление обычного int'a.
4. Из-за образовательных целей не должны использоваться stl-ные контейнеры (кроме std::string, но только как передаваемый параметр).
5. В этом задании не должна использоваться библиотека boost::operators.

Рекомендации:

1. Лучше делать базу системы счисления как можно больше. Например, uint32_t.
2. Если какая-то функция может быть реализована через public интерфейс класса, сделайте ее внешней.
3. Скорее всего, вам потребуется компиляторы g++ версии 4.5 и выше, либо MSVC 2013 для собсвтенного тестирования. Компилятор, используемый в тестах преподавателей обозначен в rules.pdf

Дополнительные усложнения (по желанию):

1. Small object optimization. **+0,5 балла при успешной сдаче.**
2. Copy on write (ленивое копирование). **+0,5 балла при успешной сдаче.**
3. Move semantics (constructor & operator=). **+0,25 балла при успешной сдаче.**

В файле lint_compilation_check.cpp приводятся тесты на компиляцию (не на корректность операций!), которые должны выполняться (если специально не указано обратного), используя ваш класс. Обратите внимание на требования наличия 2-х функций:

1. Модуль длинного числа (abs) и
2. Возведения длинного числа в короткую степень (pow). Возведение в степень должно быть быстрым (количество операций произведения длинного числа: O(logN), где N – степень).