CEng 242 Homework 4

Due 6^{th} May 2003

Some computer applications require arithmetic operations on integers without precision limits. In this homework you will implement an class for infinite precision signed integers. Number of digits in an infinite precision integer (We will name it `InfInt`) is not limited. Theoretically you can have integers with millions of decimal digits. Since they cannot fit into any primitive or static data type they should be stored and processed in dynamic data structures. Also operations like addition and multiplication cannot be carried out by the primitive operators of the processor, so a library of operators should be presented.

For all operators you can use primary school methods and algorithms. Feel free to use any algorithm as long as you understand it and you can show the internals of the algorithm when you are asked to. Your implementation should be an original work coded by you. Also another requirement is the efficiency. Adding n to 0 m times is not an efficient algorithm for multiplying m and n. Such a multiplication will last centuries for numbers with thousand digits. On the other hand it will last some seconds if you implement the primary school algorithm. Note that you are not limited to base 10 since our computers are is capable of making operations on larger integers. So you can use any base with those algorithms.
Assume the following enumerated type is defined for errors:

```
enum Error  DivisonbyZero, InvalidNumber ;
```

You will define and implement a class `InfInt` in C++ with the following member functions:

`InfInt(int)`
: Constructs and `InfInt` value from an integer.

`InfInt(const char *)`
: Given a string containing a signed decimal notation, constructs the corresponding `InfInt` object. String can contain heading and trailing spaces and negative marker '`-`' like `"  -1234567 "`. If the string contains non-digit symbols, inner spaces constructor will throw `InvalidNumber`.

`const char * toString()`
: Returns the signed decimal notation of the object as a string. The storage for the `char *` is allocated dynamically by the `IntInf` object and deallocated in the destructor.

`InfInt operator+(InfInt &)`
: A binary operator for addition of two `InfInt` objects.

`InfInt operator*(InfInt &)`
: A binary operator for multiplication of two `InfInt` objects.

`InfInt operator/(InfInt &)`
: A binary operator for integer division of two `InfInt` objects. If second operator is a zero `DivisionbyZero` is thrown.

`InfInt operator%(InfInt &)`
: A binary operator for remainder of two `InfInt` objects by integer division. If second operator is a zero `DivisionbyZero` is thrown.

`int operator{< > == <= >= !=}`
: 6 comparison operators for comparing 2 `InfInt` objects with same meanings in `int` typed values.

`ostream & operator<<(ostream &, InfInt &)`
: Not a member function but an auxiliary function to stream output of an `InfInt` value on the given output stream.

Assignment operator and copy constructor
: Assignment operator and copy constructor will be implemented such that no garbage or dangling reference will be produced

`~InfInt()` Similarly...

Your implementation should only make the requested functions public. Any other member functions that you will use will be `private`.
You should define a header file `hw4.h` containing only class definition with member variables and prototypes, and the prototype for the `operator<<(..)` function. No implementation should exist in this header file. Than, in the `hw4.cpp` file include this header file and put your implementations of member functions and the `operator<<(..)`. Your code should work in the following setting:

hw4.h

```
enum Error {DivisionbyZero, InvalidNumber};

class InfInt {
    .... // your member variables
    .... // your private member function prototypes
public:
    InfInt(int);
    InfInt(const char *);
    const char *toString();
    .... // Other public prototypes
};

ostream & operator<< (ostream &, InfInt);
```

hw4.cpp

```
// Your other includes
#include"hw4.h"

InfInt::InfInt(int a)
{
...
}
// Other private and public class definitions.
// No main() yet.
```

hw4main.cpp

```
// Your other includes
#include"hw4.h"

int main() {
   //Your trial code here
   IntInf a("12345672342342342342342342342342342");
   IntInf b("1234567234234234234234234234342");
   IntInf c(0),d(0);

   a=a+b;
   c=a*b;

   cout << a << '+' <<  b << '\n';
   .....
}
```

Then you can compile this code by:

```
g++ -c hw4.cpp
g++ -o hw4main hw4main.cpp hw4.o
```

In the submission, you should submit a `.tar.gz` file only containing `hw4.h` and `hw4.cpp`.