`CSC 330 PROJECT 3 DUE DATE: 4/23/2014

PART 1:

Design and implement a class representing a **doubly linked list**. The class must have the following requirements:

1. The linked list and the nodes must be implemented as C++ templates
2. The list must be generic – it should not implement arithmetic/logic functions.
3. It must include a destructor and a copy constructor
4. It must include methods to insert at the front and at the back of the list
5. It must include a method to return the length of the list
6. It must provide an iterator-based interface for the user from front to back
7. It must include an iterator-based interface for the user from back to front

PART 2:

The implementation of the class **LargeInt** (same idea as project 2) will use a dynamic physical structure to store the individual digits of an integer, and will provide some basic I/O and arithmetic operations that can be performed on integers.

In particular, the class should include:

1. A default constructor
2. An operator function to overload the operator +, -, / and \*.
3. An operator function to overload the operator ==
4. An operator function to overload the operator <<
5. An operator function to overload the operator >>

Note 1: since the LargeInt class does not contain pointers, there is no need for a copy constructor or a destructor.

Note 2: huge integers may be positive or negative (or zero);

Note 3: your implementation of the huge integer type must be encapsulated as a C++ class, aggregating a list object for the internal representation of the huge integer value. The huge integer type is not a list, nor does it make sense for it to be derived from a list using inheritance.

Note 4: Aside from list nodes used only within an encapsulating list template, all data members of classes and templates must be private or protected. Friend operators are permissible, but the huge integer type should absolutely not be declared a friend of the list.

Your submitted program will be assigned a score based upon:

1. whether your implementation makes appropriate use of data structures (in particular, the use of a well-designed doubly linked list )
2. whether your design shows a good object-oriented decomposition of the given problem
3. whether the internal documentation of your code is acceptable