Programming Assignment #5

This assignment gives you some practice with templated classes. The goal is to modify the *List* class from assignment #4 and create a templated class. There are only a few very minor changes from before, so it shouldn't require a lot of time. The templated class just allows you to construct lists of different types (e.g. integers, doubles, StopWatches, Students, etc.) Each node (*Node*) in a list contains a data member and a pointer to the next *Node*. This is the partial interface to the class:

```
template <typename T>
class List
{
  public:
    List();                 // Default constructor
    List(const List &list); // Copy contructor

      // Operator for printing lists (<<)
    friend std::ostream &operator<< <T>(std::ostream & os, const List<T> &list);

      // Other public members

  private:
    struct Node
    {
      Node(T value);          // constructor (conversion)
      ~Node();                // destructor
      Node *next;             // pointer to the next node in the list
      T data;                 // the data stored in the node (generic type)
      static int nodes_alive; // count of nodes still allocated
    };

    Node *head_; // pointer to the first node in the list
    Node *tail_; // pointer to the last node in the list
    int size_;   // number of items on the list

    Node *new_node(const T& data) const; // allocate node, initialize data/next
};
```

There are 18 functions that need to be implemented. (A few of them are already implemented.) Many of the functions will call other functions you've implemented (code reuse), so the amount of code is not that great. The "worker" functions are `push_back`, `push_front`, and `pop_front`. Once these functions are implemented and **thoroughly tested**, the rest of the assignment is fairly straight-forward. The sample driver shows many example function calls with the appropriate output. You should be able to glean all of the information required from the driver.

Since almost all of the code is re-usable from a previous assignment, failure to get the correct output will result in a poor grade. By now, all of you should be very comfortable with pointers and linked lists, since we've spent so much time with them. If your previous code was broken, please do not simply "templatize" that broken code and resubmit it. I will expect you to fix all of the problems before resubmitting it. (This means you'll need to fix incorrect behavior, memory leaks, poor coding, documentation, etc.)

Other criteria

1. You will only need to make some minor modification to your previous List assignment. All of the rules regarding that assignment are applicable to this assignment as well.
2. You must allocate the nodes using **new.** The only function that should use **new** is new_node, which you are given. This means that the keyword **new** should be used exactly once in your program.
3. Only the push_front and push_back methods should call new_node (to create nodes).
4. You must deallocate the nodes using **delete**. The only function that should use **delete** is pop_front. This means that the keyword **delete** should be used exactly once in your program.
5. You are given the implementation for **`operator<<`** in the .cpp file.
6. **New behavior:** We won't be keeping track of how many Lists have been created. We will only be tracking how many *Nodes* have been created. This code is provided.

7. **New behavior:** Since this is a templated class, you must include the .cpp file at the end of the .h file exactly as shown in the header file that is posted.

Deliverables
You must submit the header file (List.h) and the implementation file (List.cpp). These file must be zipped up and submitted to the appropriate submission page.

Source Files	Description
List.cpp	The implementation file. All implementation for the functions goes here. You must document the file (file header comments) and all functions (function header comments) using the appropriate Doxygen tags. You must properly document all functions, even if they were provided for you.
List.h	The header file. You may need to modify the one I posted. **No implementation is permitted in this file.**

Usual stuff
Your code must compile (using the compilers specified) to receive credit. The code must be formatted as per the documentation on the website.

Make sure your name and other info is on all documents.