CSCI 235 - Spring 2015
Homework 4

Due: Wednesday, May 6th, 2015, 11:59 pm

1 Linked Queues

You are given an incomplete implementation of the LinkedQueue ADT.

The header file `LinkedQueue.h` is complete and there is no need to change it. You have to add code in some functions of file `LinkedQueue.cpp`.

You have to add code between the comments
`//ADD NEW CODE`

`//END ADD CODE`

Specifically add code in the following order to the given functions:

1. `CopyNodesFrom()`

   This function implements a deep copy from a given queue. You need to add some code in the while loop.

2. `operator=(...)`

   This is the implementation of overloading the assignment operator.

3. `operator<<(...)`

   Friend overloading of operator `<<`. In C++, an overloaded operator cannot be a member of a class if the left hand side is not the class itself. Here, we want to overload the `<<` operator so that we can print the contents of the queue to a stream. `cout` is an object of the class `ostream` and the class `ofstream` inherits from `ostream`. Therefore, you will implement the following function:

   `ostream& operator<<(ostream& out_stream, const LinkedQueue<FriendItemType>& output_queue)`

   So this function should work in all scenarios where we have output streams. Two examples follow:

```
LinkedQueue<string> queue1;
queue1.Enqueue("hello");
queue1.Enqueue("world");

out << queue1; //This should print [hello, world] to standard output.

ofstream myfile("some_file");
myfile << queue1; //This should print [hello, world] to the file "some_file".
```

Note that we want to print the contents of queue separated by commas and within "[" and "]".

4. `operator+(...)`

   Overloading of the + operator. By + we mean concatenation of two queues. So if

```
LinkedQueue<string> queue_1; // Enqueue/Dequeue items
LinkedQueue<string> queue_2; // Enqueue/Dequeue items
LinkedQueue<string> queue_3 = queue_1 + queue_2;
```

   then `queue_3` will be the concatenation of queues `queue_1` and `queue_2`.

Note that the code will compile in the state it is right now, but it will not run. After you implement all functions, the main() function will test them, but you do not have to modify the main.cpp file. You can (and encouraged) however to comment parts of the main.cpp code during testing of your implementation.