CS 215 – Fall 2018
Lab 9

Learning Objectives:

- implementation of (part of) a Linked List class.
- use of destructors to "clean up" dynamic memory.
- use of a class with "inline" methods.

General Description:

You are to implement a "buddy" list app (since *friend* is a reserved word in C++) using a Linked List. The list only has two actions it can perform: *add* a new buddy and *print* the list. Other actions will be added for Project 4.

Specifications:

In MSVS, start a new project for Lab 9. Copy the two .h and two .cpp files from the course website and add them to your project. **Note: there will be no buddy.cpp for this lab. See buddy.h for details.**

Finish the code by completing the **// TODO:** sections in the code:

main.cpp

Put your name and section in the comments

buddy.h

Note this class uses *inline* methods, where the interface and implementation are all written in the class declaration.

- make class buddyList a friend of class buddy
- add a destructor. If the next pointer is not NULL, delete the next pointer. Include the cout given in the comments in your code.

buddyList.h

- add the prototype for the class destructor

buddyList.cpp

- constructor: sets the head to NULL and prints the message in the comments.

- destructor: when the list is not empty, delete the head. Print the message in the comments.

- add(strings): allocate a new buddy node. Set the arguments given in the new node. Invoke the second add() to add the new node to the list.

- add(pointer): given a pointer to an already allocated and populated node, add the new node to the head/top of the list. This is only 2 lines of code [hint: see stack::push() in the Stacks and Queues notes)

- print(): add a loop to invoke print() in each node in the list.

Submission:

Demo to your TA. Zip all .cpp and .h files (and nothing else) into a .zip file and submit in Canvas.

Sample Execution: