**CPSC 1071   
Lab 10 - C++ Singly Linked List**

**Objectives**

Earlier this semester we implemented a linked-list and iterator modules in C. This week, you will be implementing a C++ version a linked-list and iterator. Many of the concepts will be the same but will be restructured to fit into the C++ model and structure of programming.

**Task #1: implement the linked list in C++**

**The Node class**

class Node {

public:

Node();

Node(void\*);

~Node();

Node\* getNext();

void\* getItem();

void setNext(Node\*);

private:

void\* item;

Node\* next;

};

Above, the C++ version of the Node class is very similar to the *node* struct that we used in the linked-list lab earlier this semester. Now, we are able to associate methods along with member variables with a Node class.

**The List class**

class List {

public:

List();

~List();

void addFirst(void\*);

void\* getCurItem();

Node\* getHead();

void reset();

private:

Node\* head;

Node\* current;

};

Again, the list is very similar to the list struct that we used in the C version of the linked-list.  
  
**Note**: look for the differences in the C and C++ versions of the source code. We no longer have *list\_* prefix for our functions since we can now associate methods along with member variables using a class. We are also taking advantage of C++'s use of constructors. This means that we do not need to use *malloc* or *new* within our constructor. We only need to initialize our member variables.  
  
**Note**: This does not mean we will not be using the *new* operator however. Instead of using *malloc* to create a new node struct pointer, we will be using the *new* operator to create a new instance of a Node pointer (Hint: pay attention to the constructors that we can use for the node; we can use the specialized constructor to initialize our item directly).

**Task #2: implement iterator in C++**

The Iterator class

class Iterator {

public:

Iterator();

Iterator(List\*);

~Iterator();

void advance();

bool hasNext();

Node\* get();

private:

List\* list;

Node\* current;

};

With the new Iterator class, we see one particular change to the *l\_hasnext()* function we saw in the C version of the iterator. We have changed this to the *hasNext()* method here and return a *bool* type. The *bool* type is a C++ type that can give a *true* or *false* value instead of using 0 or 1 that we used previously.

**output**

\*

\*\*

\*\*\*

\*\*\*\*

1

2

3

4

5

6

7

8

removing list item

removing list item

removing list item

removing list item

removing list item

removing list item

removing list item

removing list item

removing list item

removing list item

removing list item

removing list item

**Getting the file for today's lab**

Use

lab1071copy  10

to copy the .tar files for today's lab

to untar your lab use tar -xvf lab10.tar. if you need the C version, then use tar -xvf c\_list.tar

**Handin**

When you are finished and have commented and compared your output with the **output.txt** file provided, ***make clean*** and tar up your **.cpp** and **.h** and **Makefile**:

tar -cvf mylab10.tar  \*

Use [Handin](https://handin.cs.clemson.edu/) to turn in your mylab10.tar