Linked Lists in C++

Lab 11

Wednesday, April 17, 2013

CSC 062: Spring, 2013

In this lab, we will be playing with pointers by building a linked list class similar to our Java implementation. In addition, we'll also look at a few experiments to understand how memory needs to be managed in Java.

Getting started

Copy all of the contents from:

```
common/cs/cs062/labs/lab11
```

On the course web page, look at our notes for lists and refresh your memory on how we implemented linked lists in Java. Take a look at the definition of the `node` class in C++ (both the .h and .cpp) files. What is different?

Compile the `node` class by typing:

```
g++ -c node.cpp
```

Notice that we're just compiling the file, but not building an executable yet.

LinkedList

Now, take a look at the linked list header file (`linkedlist.h`). The header file contains a basic set of linked list methods. Implement these methods in a file called `linkedlist.cpp`. Most of these methods should be a straightforward translation of the code from Java. Note, however, that you will need to use pointers! Just to keep you in the good habit, `#include <cassert>` and use assert statements appropriately in your code.

Again, to compile this type:

```
g++ -c linkedlist.cpp
```

Using the linked list class

Once you have it compiling and you think you have it working, look at the `linkedlisttest.cpp` class, then compile it:

```
g++ -c linkedlisttest.cpp
```

and then compile/link all of your previously compiled object files into an executable binary (notice that `linkedlisttest.cpp` has a `main` method, which is required to construct an executable):

```
g++ -o linkedtest node.o linkedlist.o linkedlisttest.o
```

or

```
g++ -o linkedtest *.o
```

Run the test:

```
./linkedtest
```

If all works well, you should see the numbers from 0 to 9 printed out, except 4 is replaced by 100.

Note that anytime you change a .cpp file, you'll need to recompile that particular file, but then also relink the executable with the -o command.

Make

Hopefully, the act of changing and recompiling your code with the above instructions has convinced you of the need for a better solution. Luckily, a better solution exists.

In a previous lab, we asked you to read the "highly abbreviated introduction to command-line tools" on the class webpage. You should re-read the section on make again before continuing.

We have given you a non-working Makefile as part of this week's lab files. You should be able to modify the Makefile by adding the commands you used above in the indicated places. When you are done, test your new Makefile by typing "make clean" on the command line and confirm that your object (ends with .o) files have been removed, as has your executable. Next, type "make" on the command line, and make sure your project has been built correctly.

What to submit

Perform the following exercises, and answer the associated questions. Place your answers in a text file named "lastname-Lab11.txt," or "lastname1-lastname2-Lab11.txt" if you worked with a partner, where you have replaced *lastname* with your last name. Place this file, along with your source code, and your Makefile, in a folder named "lastname-Lab11" and submit this folder to the dropbox.

1. Change the `main` method in `linkedlisttest.cpp` to run `test2` and recompile. Repeat for `test3` (it's a little weird to store a `Node` in a `vector` since we're ignoring the `nextElement` pointer, but I wanted to convince you that `test2` and `test3` are roughly the same).

   Do these results surprise you? Why are these results different?

2. Change "`LinkedList l2 = l`" to "`LinkedList l2(l)`".

   Does this change your result? Where does that constructor come from?

3. Change the linked list variables in `test2` to be linked list pointers. Use the "`new`" operator to create a new linked list object for `l`. Set `l2 = l`.

   Does this change your answer? Does this make sense?