Computer Science 62
Lab 12

Wednesday, April 23, 2014

This lab examines C++ memory management and will experiment with a linked list class called `SimpleList`. As usual, you can work with a partner for this lab.

Getting started

Begin by making a directory for Lab 12 and copy the files from `/common/cs/cs062/labs/lab12`. Look at the different files and figure out what is going on with the code. Note that the `SimpleList` class shows an example of static instance variables. The instance variables are declared in the header file with the `static` keyword (just like Java) and they are initialized at the top of the `SimpleList.cpp` file. We refer to them using the full name, for example `SimpleList::countDefConst`.

Compile all of the files, generate a final binary and then run it (look at the section at the end of last lab for how to compile multiple files in C++).

The program prints the number of calls to each constructor and to the destructor. The total number of constructor calls is equal to the number of destructor calls. Locate where each call is made. In particular, identify the source of the (large number of) copy constructor calls. Make sure you understand why you're getting the results you're getting before moving on.

Pointers

The function `test0` uses objects, not pointers. Now let's try it with pointers. In your `SimpleTest.cpp` file, make a copy of `test0` and call it `test1`. (We ask you to copy and rename so that you can revisit your old, unmodified code at another time. Create experimental functions `test2`, `test3`, ... freely.) Change the declarations in `test1` of `lst` and `tmp` to pointers.

```
SimpleList *lst = new SimpleList();

for (int i = 0; i < 6; i++) {
  SimpleList *tmp = new SimpleList(i, *lst);
...
```

Then change the method calls to match the pointer types. Explain the sources of any errors you encounter. When you run the program, is there a difference in the numbers of constructor and destructor calls? If so, where does the difference arise? Is it possible to bring the calls back into balance without changing the underlying class?

Making Errors

We are guarding against three types of errors: dangling pointers, memory leaks, and double deletion. One way to understand the errors better is to intentionally make them. Add more test functions to your program and write code that makes each of the errors. (Again, add the functions; do not replace the old ones.)

What to hand in

If you worked with a partner, put both of your names somewhere in the comments at the top of the `SimpleTest.cpp` file. Create a directory called "Lab12_LastNameFirstName". Copy all of the files (`SimpleList.cpp`, `SimpleList.h`, and `SimpleTest.cpp`) into this directory. Drag this directory to the CS62 dropbox.