UNIVERSITY OF WISCONSIN
Computer Sciences Department

CS 537
Fall 2018

Barton Miller

Programming Assignment #3-1

Handed out: Wednesday, November 7
Due: Wednesday, November 14 at 5pm

The goals for this assignment are:

A. Learn about an important type of tool that can check your program for memory errors. In particular, you'll get experience with **valgrind**.
B. Review the code that you wrote for Program 3 to see what kind of undetected mistakes you might have made.
C. Learn how to understand and fix the mistakes reported.

Your four tasks for this assignment are:

1. Make sure that your Program 3 solution compiles and links ("builds") with `-Wall` and `-Wextra` with **no warnings**, and scans clean with the Clang Static Analyzer. Presumably, you already did this step when you completed Program 3. If you have not yet done this step, then you'll need to modify your program **to eliminate the warnings**.
2. Learn how to use valgrind (which is super easy).
3. Learn to interpret the output of valgrind and use it to fix any problems that you find in your code.
4. Think abouts the results and be ready to discuss in class whether you considered any identified issues seemed to be worth fixing.

If you have questions as to what the messages mean or problems running CSA, talk with a TA or Bart.

Running Valgrind

There is a great Quick Start Guide to valgrind, which will tell you almost everything you need to know about running it.

You should also take a bit of time to look over the full Valgrind User Manual.

Valgrind supports several checking tools; the one that we will use is Memcheck. If you do not specify which tool to run on the command line, then it will default to running Memcheck (so you **do not** need the `--tool` option).

Firstd, make sure to compile your program with `-g` and `-O0`.

You will run valgrind with the following options:

```
--leak-check=yes
```

Check for memory leaks, i.e., memory that is allocated but never freed. Note that such leaks are not necessarily bugs. If the leak was in the allocation of your read buffer each time, then that would be a concern.

`--track-origins=yes`

Provides more detailed information about references to unitialized memory.

`--read-var-info=yes`

Provides more detailed information about references to illegal addresses.

So, your command line to run valgrind on your program might look something like:

```
valgrind --leak-check=yes --track-origins=yes --read-var-info=yes 537make depgraph.o
```

You'll need to update your code to fix any **errors** that valgrind finds. Note that you do not need to fix errors that come from a leak of memory that was allocated but does not need to be freed before exit.

Testing

You'll want to run your program under valgrind with all of your various test input files. Note that valgrind, with the above options, will cause your program to run **much much** slower. So, for big files, you'll need to start the program running and do something else for a while.

Deliverables

You will work with your original partner for Program 3 and, again, turn in only one copy of the output. You will also turn in the modified copy of your code, if there are were changes needed to file memory errors.

You'll put the output of valgrind in a file called **`valgrind.txt`**.

Also, in your README file, you will describe which lines of your program needed to be changed This will be a list of (file name, line number) pairs, with a comment about what was the error at that line and how you fixed it.

Handing in Your Assignment

Your CS537 handin directory is `~cs537-1/handin/`*`your_login`* where *`your_login`* is your CS login. Inside of that directory, you need to create a `proj3-1` subdirectory.

Copying your files to this directory is accomplished with the `cp` program, as follows:

```
shell% cp *.[ch] makefile README ~cs537-1/handin/your_login/proj3-1
```

You can hand files in multiple times, and later submissions will overwrite your previous ones. To check that your files have been handed in properly, you should list `~cs537-1/handin/`*`your_login`*`/proj3-1` and make sure that your files are there.

When you are working in a pair, you should:

1. Submit only one copy of your code and other files.

2. Make sure that both of your names and NetIDs are in comments at the top of each source file.
3. Both of you should create another file called partner.txt in **both** of your `proj3-1` directories, where you should have two lines that have each of CS login and netid of you and your partner.

Original Work

This assignment must be the original work of you and your project partner. Unless you have explicit permission from Bart, you may not include code from any other source or have anyone else write code for you.

Use of unattributed code is considered plagiarism and will result in academic misconduct proceedings (and "F" in the course and a notation on your transcript).

Last modified: Wed Nov 7 12:49:44 CST 2018 by bart