Princeton University
COS 217: Introduction to Programming Systems
The Valgrind Tool

What is it?

Valgrind is a tool to help you analyze your application's dynamic memory management. It can help you find memory leaks, multiple frees, and dereferences of dangling pointers. It may help you find other dynamic memory management errors as well. Valgrind is an open-source tool that has been developed by multiple programmers over multiple years. Browse to http://valgrind.org/ for details.

How do I use it?

Suppose you wish to use Valgrind to help you debug an application named myapp. Further suppose that myapp consists of source code files mysourcecode1.c and mysourcecode2.c. Assuming that you've configured your CourseLab programming environment as described in the first precept, follow these steps:

(1) Use the **gcc217** command with the **-g** option to preprocess, compile, and assemble mysourcecode1.c and mysourcecode2.c:

```
gcc217 -g -c mysourcecode1.c
gcc217 -g -c mysourcecode2.c
```

(2) Use the **gcc217** command with the **-g** option command to link mysourcecode1.o and mysourcecode2.o, thus creating executable file myapp:

```
gcc217 -g mysourcecode1.o mysourcecode2.o -o myapp
```

Note that steps 1 and 2 can be combined by issuing a single command:

```
gcc217 -g mysourcecode1.c mysourcecode2.c -o myapp
```

(3) Execute myapp through the **valgrind** command, by typing "valgrind" followed by your program's name (and command-line arguments, as appropriate):

```
valgrind myapp arg1 arg2 ...
```

Valgrind writes messages to stderr. It does so as your program executes, so Valgrind's messages often are interspersed with your program's output. If Valgrind writes many messages, then you might find it convenient to redirect stderr to a file, and then examine the file subsequently:

```
valgrind myapp arg1 arg2 ... 2> valgrindreport
cat valgringreport
```

Valgrind is a powerful tool; its intended audience is experienced C programmers. So newcomers to C might consider its output to be cryptic.

Generally, you should read Valgrind's output from bottom to top. Doing so provides a function-call trace from your `main()` function, perhaps through multiple levels of function calls, to the function that was executing at the time Valgrind detected a problem. That information often is helpful, even if the error is elsewhere in your code.

Feel free to contact your preceptor if you have trouble interpreting the output of Valgrind.