15-122: Principles of Imperative Computation, Summer 2012
Assignment 0: Getting Started

Jamie Morgenstern (jamiemmt@cs)*

Out: Monday, May 21, 2012
Due: Tuesday, May 22, 2012

Welcome to 15-122, Principles of Imperative Computation, Summer 2012 Edition! This low-mark (10 points) "zeroth" homework is designed to give you a chance to get used to the course tools relevant to your success in 15-122 in a low-stress manner. Although it is not for many points, we encourage you to complete it so you can get used to some of the course workflow and work out any kinks before the first graded assignment.

1 Prior to doing anything else

Read and follow the directions here:

`http://www.cs.cmu.edu/~jamiemmt/teaching/su-122/recitations/recitation01.html`

2 Written: Piazza Bulletin Board (1 points)

Many course-relevant communiqués will be posted to the course *bulletin board*, or *bboard* (not to be confused with *Blackboard*, the web-based software we use for recording grades and administering quizzes). The bboard located here:

`http://piazza.com/class_profile#summer2012/15122`

Exercise 1 (1 pts)**.** Go to the Piazza interface. Find the post entitled, "Homework 0: introductions" and respond to it with a short introduction of yourself, including your name, your major, and an answer to one of the following questions, your choice:

1. Who's the computer scientist you most admire?

*Previous Versions of this lab by William Lovas and Erik Zawadzki

2. What's your favorite book or movie featuring computers or computer science in a significant way?

3. What do you hope to learn from this course?

3 Programming: Exponentiation (4 points)

To get some experience writing and submitting C_0 code, we'll write the canonical "Hello, World!" program (and one more tricky program) and submit them through our interface.

Exercise 2 (3 pts)**.** Type the following main function,

```
#use <conio>
#use <string>

int main() {

  print("Hello, world!\n");
  return 0;
}
```

which prints to the screen "Hello, World!" into a file `hithere.c0`. In order to run your code, you must first compile it:

```
cc0 -d hithere.c0 -o hello
```

where `-d` tells the compiler to check the contracts of your code, and `-o hello` tells the compiler what to name the executable it outputs. Now, to run the executable, type

```
./hello
```

Now, submit your two files, `hithere.c0` and `hello`:

```
handin -a hw0 hithere.c0
handin -a hw0 hello
```

Exercise 3. Now, we want you to write more canonical C_0 code, so please write a simple function `int fib(int n)` to compute the nth Fibonnaci number, into a file `fib.c0` , including appropriate `//@`-annotations to document the function's pre- and post-conditions, loop invariants, and any assertions.

Be sure to test your code, but do not include a `main()` function in your submitted file. Instead, test your code by writing a `main()` function in another file, say `fib-testing.c0`, and compile both files together, e.g.,

```
cc0 fib.c0 fib-testing.c0
```

or

```
cc0 -d fib.c0 fib-testing.c0
```

(Remember that the `-d` flag enables dynamic checking of annotations.) Alternatively, you may test your code using the `coin` interpreter.[1]

Exercise 4 (1 pts)**.** Create a file `README.txt` explaining your code and its invariants briefly. Please submit your files:

```
handin -a hw0 fib.c0
handin -a hw0 README.txt
```

4 Written: Linux Commands (5 points)

Unix-like systems, such as Linux, are common and important to know. This set of exercises is designed to help you become familiar with using a set of common Linux commands.

Here is an incomplete list of important commands and concepts, some of which will be necessary for this section:

- `man`, the most important Linux command
- File-system commands: `cp`, `mv`, `cd`, `ln`, `mkdir`
- Text display commands: `cat`, `tac`, `wc`, `head`, `tail`, `echo`, `less`, `sort`
- Search commands: `grep`, `find`
- Remote commands: `ssh`, `scp`
- Permissions and user groups commands: `chmod`, `chown`
- Redirection and piping: <, >, >>, >&, |

In all of the follow exercises, we encourage you to test out commands, figure out their behavior, and make sure that your answers work.

Assemble all of your answers into a single text file called `linux_ans.txt`

Exercise 5 (1 pts)**.** Suppose you just coded, successfully compiled, and tried to run a program called `foo`, and your program gave you the following error:

[1] We'll show you how in Tuesday's recitation.

```
[Your Command Prompt]% ./foo
bash: ./foo: Permission denied
```

What does this error mean, and what command would you use to fix it (i.e. get your program to run correctly)?

Exercise 6 (1 pts)**.** Suppose that `head`—a command that prints the first few lines of a file—is broken on your system. Give a single command (i.e. you can use redirection and piping of existing commands, but not a script) that provides the same function as `head` on the file `my_file.txt`.

Hint: you will probably have to pipe output—consider using `|` somewhere in your command.

Exercise 7 (2 pts)**.** Your friend wrote a program that keeps a number of different logs, all with the `.log` suffix. These logs are kept in many different subdirectories of `$PROGRAM_PATH` (this is a shell variable). All logs have the same `[DATE];[COMMENT]` format, where '`;`' is the *delimiter* of the column[2].

By *subdirectory* we mean any directory in the *directory tree* under the root directory `$PROGRAM_PATH`, including itself. So `$PROGRAM_PATH`, `$PROGRAM_PATH/src/`, and `$PROGRAM_PATH/examples/soy_late_1/` are all considered to be subdirectories for the purposes of this assignment.

For example, the first few lines of of `$PROGRAM_PATH/bin/engine/tmp/init.log` are:

```
Sat Jan 14 07:05:47 EST 2012;; Program started, building heap...
Sat Jan 14 07:05:48 EST 2012;; Heap build finished.
Sat Jan 14 07:05:52 EST 2012;; Kicking off monitoring daemon...
```

Notice that, like in most logs, the entries are in chronological order. In a single Linux command (possibly with redirection and piping), how would you combine all these logs into a single chronologically-ordered log named `$PROGRAM_PATH/master.log`? You might find the commands `find` and `sort` useful here.

Exercise 8 (1 pts)**.** Your friend made some some changes to the program mentioned in question 7, and now it is no longer working. The program, when asked to run, reports the following error:

```
[ERROR] Class 1 error! SELECTED_SYRUP set to WALNUT_SPICE.
[...]   This is incomptabile with the current DRINK_MODE.
```

You have no idea what this mean, but suspect that it has something to do with one the program's many configuration files. Unfortunately, unlike the log files, the configuration files do not have the same suffix, and could be any non-binary file.

[2] If you do not know what a delimiter is, this would be a good thing to look-up. They're everywhere.

To investigate this strange error, please write a single command (possibly with redirection and piping) that prints every text file line that contains the string "`WALNUT_SPICE`" and prints no other lines. In particular, your commands should not print any lines at all from any of the binary files. Your command should check every text file in any subdirectory of `$PROGRAM_PATH`.

Now, hand in all of this exercise:

```
handin -a hw0 linux_ans.txt
```

Once you have completed these exercises, you should be prepared to tackle homework 1, image manipulation!