COMP 211: Lab 1
Fall, 2014

1 Installing software

You will need three different pieces of software for the course:

1. *Unix* is a computer operating system (it plays the same role as Windows or Mac OS) from the 1970s; a modern variant called *Linux* is widely-used by programmers and on servers. It is very common for programmers to use a *Unix-style terminal* to interface with their computer. This is a text-based interface for doing the same kinds of things you would do by pointing and clicking in Mac OS or on Windows, like making new files or folders, moving files around, etc. However, because it is text-based, you can write programs to do sequences of commands—we will use this later in the course.

   If you are using a Mac, you already have a Unix-style terminal; just open `Utilities/Terminal`.

   If you are using Windows, you need to install a Unix environment called Cygwin.

2. The C0 programming language. This is the particular programming language we will use for this course.

3. Programs are just text files (sequences of letters and numbers and punctuation). You can in principle use any text editor to edit programs (Notepad, Notepad++, TextEdit, ...). However, the process is **much** easier if you use a text editor that knows about programs, so we will require you to use one of:

   - Emacs: One of the traditional Unix text editors. Pros: completely free; very powerful once you know how to use it. Cons: You will have to learn how to use it; the interface and keyboard shortcuts are different than what you're used to.
   - Sublime Text: A new text editor designed for programmers. Pros: Some modern features such as graphical auto-complete; interface will be more familiar to you at first. Cons: Free to use an unregistered version, but will often ask you to register it; just ignore this.

Task Follow the instructions on the resources tab of the course web page

`http://dlicata.web.wesleyan.edu/teaching/pic-f14/resource.html`

to install a Unix-style terminal, C0, and a text editor.

2 Some Basic Unix Commands

Open up Terminal (Mac) or Cygwin (Windows).

1. Type

   ```
   pwd
   ```

   (short for "print working directory"). This prints out the name of the directory/folder you are currently in.

2. Type

   ```
   ls
   ```

   (stands for "list") This shows you the contents of the current directory/folder.

3. Type

   ```
   mkdir new-folder
   ```

   This creates a new empty directory/folder named `new-folder`.

4. Type

   ```
   ls
   ```

   You should now see `new-folder`.

5. Type

   ```
   cd new-folder
   ```

   (short for "change directory"). This moves you inside `new-folder`.

6. Type

   ```
   ls
   ```

   This shows the contents of the directory you are currently in, which is now `new-folder`, which is empty.

7. Type

   ```
   touch test.c0
   ```

   This creates a new empty file named `test.c0`.

8. Type

```
ls
```

You should now see `test.c0`

9. Type

   ```
   cp test.c0 test2.c0
   ```

   This copies the file `test.c0` to `test2.c0`.

10. Type

    ```
    ls
    ```

    You should see both `test.c0` and `test2.c0`.

11. Type

    ```
    mv test2.c0 test3.c0
    ```

    This moves (renames) the file `test2.c0` to `test3.c0`.

12. Type

    ```
    ls
    ```

    You should see both `test.c0` and `test3.c0` but not `test2.c0`.

13. Type

    ```
    cd ..
    ```

    This means "change to the directory *containing* the folder I am currently in."

14. Type

    ```
    ls new-folder
    ```

    This lists the contents of the specified folder `new-folder`.

3 Running C_0

In the terminal, type

```
coin
```

This starts the program `coin`, which is what we will use to run C0 programs. You should see

```
C0 interpreter (coin) 0.3.2 'Nickel' (r349, Wed Aug 28 18:31:41 EDT 2013)
Type `#help' for help or `#quit' to exit.
-->
```

The `-->` means C0 is waiting for you to write a program for it to run.

Type

```
1 + 2;
```

and press enter. You should see

```
3 (int)
```

This means "the program 1+2 produced the answer 3". You should also see another line

```
-->
```

which means that coin is waiting for you to write more programs.

You've written your first C0 program!

Type

```
3 * 5;
```

and press enter. What does `*` do?

Type

```
6 / 3;
```

and press enter. Then type

```
7 / 3;
```

and press enter? What does `/` do? Try a few more examples to confirm your guess.

Type

```
#quit
```

This exits coin.

4 Unix shell vs. C0

Here is something you might get confused by: We use the Unix terminal both to interface with the operating system (e.g. to make new folders and copy and move files) and to run the program `coin`, which we were using above to execute C0 programs (e.g. to type in `1+2` and have it compute `3`). We use both of them in the same way: you type in things (`ls`, `1+2;`), and then something happens.

However, it is important to keep in mind that **the things you can type in when you start Terminal/cygwin are different than the things you can type into coin**. E.g. when you just start the terminal it doesn't understand `1+2;`

```
$ 1+2;
-bash: 1+2: command not found
```

The error "1+2: command not found" means it doesn't know what to do with what you typed.

After you start `coin`, it doesn't understand `cd new-folder`:

```
--> cd new-folder
<stdio>:1.1-1.7:error:consecutive expressions
```

The error message ("error connsecutive expressions" means C0 didn't even understand how to make grammatical sense of what you wrote).

In fact, what is happening is this:

- When you start terminal/cygwin, it runs a program called the *Unix shell*, which can run programs like `cd` and `ls` and is good for working with files. *The shell is itself a programming language*, so you've really written programs in two different languages, all in the first lab!
- Inside the shell, we run a program called `coin`, which itself runs C0 programs.

In instructions, we will write `$` (meaning shell) and `-->` (meaning coin) to indicate whether you are supposed to type something into the shell or into coin:

```
$ type this into the shell
```

```
--> type this into coin.
```

(So when we show instructions like this, don't type the `$` and `-->`).

5 Libraries

In the shell, start `coin` again:

```
$ coin
```

Task 5.1 What happens when you run the following?

```
--> string_join("i am " , "the walrus");
```

Some functions are in *libraries* that you need to load separately.

Task 5.2 Quit coin by typing `#quit`, and then start it using

```
$ coin -l string
```

to load the library named `string`. Then run the above command again.