Assignment 1 Instructions
CISC 220, Fall 2018
(updated Sept 19)

Important Administrative Notes – Please read these before you start the assignment! You may lose points if you ignore them.

Deadlines: The deadline for this assignment is **8 am on Tuesday, September 25**. The submission area will stay open for an extsra 24 hours so you can submit until 8 a. m. on Wednesday, September 26 – but if you submit during that extra day there will be a 10% deduction in your mark. (In other words, we'll mark your assignment as usual and then multiply your mark by 0.9 as a penalty for lateness.) After 8 a.m. on Wednesday we will not accept any more submissions.

If you are sick or have another valid reason for handing the assignment in during the grace period or not being able to hand it in at all, make a submission to the Academic Considerations Portal to document this. If your submission is approved I will contact you by e-mail and either give you an extension or modify your marking scheme to ignore this assignment.

Administrative Penalties: You **must follow the instructions** about each script. In particular, use exactly the file names requested and make sure the scripts that need arguments take the number of arguments specified in the order specified. If you ignore the instructions it will add to our testing time and so there will be a 10% penalty for this, even if your script essentially does the right thing.

If you are late *and* don't follow the instructions there will be a 20% penalty.

This and every other assignment in CISC 220 is to be done individually. **Group work for marked assignments is not allowed.** You may not work with other students but must hand in your own work. If you need help, talk to the prof or a TA. Do NOT show your work to other students or look at work done by someone else. Please review the material about Academic Integrity in the course syllabus and be warned that I will report unauthorized collaboration to the Dean, which may result in a penalty quite a bit worse than a zero on the assignment.

You must hand in your assignment files to OnQ in the **"Assignment 1" Dropbox area** (under the "Assessments" menu in OnQ). We will not accept assignments in any other form – on paper, via e-mail, on flash drives or anything else. This is a large class and we need to keep the submissions organized in one place.

For this and all assignments, you may lose points if you don't **follow the instructions carefully** and observe the restrictions given. Please understand that this is a large class and we can't afford the time to make allowances for lots of individual variations. With more than 200 students in this class, a minute or two to adjust the testing procedure for each one adds up to more extra time than we've got in our budget! Also, being precise is part of programming. If you're part of a programming team and you're asked to write a module as part of a larger project there will probably be rigid specifications about function names, parameters, etc. If you write a module that does not follow those specifications it won't fit with the rest of the project even if it does pretty much the right things. Your supervisor and co-workers will be quite unimpressed and you will be asked to fix these issues immediately.

A note about file names: I personally prefer not to put extensions on shell script names and I generally don't use extensions on the names of the scripts I write in class. But we're stuck with the limitations of OnQ and OnQ insists that all files uploaded to a Dropbox must have an extension. For example, if I create a script that counts something, I would just call it `count`, but I can't ask you to do that for assignments; Dropbox wouldn't take it. So in this assignment and in Assignment 2 I'm asking you to put the extension `.sh` on your script names – as in `count.sh`. instead of just `count`. This is an extension some Linux programmers like to use, so even though you won't see me using extensions for my examples in class it's still something you'll see in the Linux world.

This is an individual assignment! You are expected to do your own work. Collaborating with other students or copying work from another person (whether a student or not) is unacceptable and can lead to academic integrity charges. Such charges may result in penalties worse than a zero on the assignment and a permanent note in your academic record. It is not worth it!!! And furthermore, handing in work done by others doesn't help you learn. The real goal of this assignment and all of the assignments in this course are to help you learn and practice the course material. If you don't do this you will lose marks in your quizzes and final exam, which are worth a lot more points than the assignments.

Getting Help: If you have trouble with this assignment, there are ways you can get help that are compatible with academic integrity:

- Visit my office hours or talk with me after class. My office hours are always posted on my home page: http://research.cs.queensu.ca/home/malamb/. Any changes to them will be announced on the OnQ news.
- I am currently trying to organize one or two TA "help sessions" in addition to my own office hours. These would be times when there will be a TA on duty in one of the CASLab location to provide assistance ad you work on your assignment. They will not do your assignment for you, but they can help you debug and give suggestions. I will announce details on the OnQ news.
- You may send mail to me but with a class this size I can't type long essays so this is not an especially good way of getting debugging help. If you want me to look at a script, please send it as an attachment so that our e-mail programs don't mangle it, and please ask a specific question. In other words, if you ask "why doesn't this work?" you won't get an answer. Please ask a specific question – for example, "when I call the script with *<some particular set of arguments>* I get *<this incorrect answer>*. Or "when I try to run this script with *these arguments* I get the following error message that I don't understand". I will try to give you a hint if my time permits, although I won't dictate code for fixing the problem. I can't guarantee to answer messages sent to me less than 24 hours before the due date.

OK, after all of that, here is your assignment! It consists of 3 scripts that you must write and submit. I consider this a fairly basic assignment to get you started with Linux. Assignment 2 will ask you to write more difficult shell scripts.

Script 1: Varying the "ls" command

This script is an exercise in reading "man pages" to learn about command options. This is an important skill for any Linux programmer because most commands have lots and lots of useful options.

We've talked about the `ls` command, which lists files and directories. The command `ls` with no arguments will print the names of all of the files in the current directory in alphabetical order with multiple file names per line, arranged in columns. For example, if you are in the directory called `/cas/course/cisc220/poems`, the output of `ls` will look something like this:

```
bed      gentle    jabberwocky  michaelis  road    tolls    will
birches  gustibus  letter       nov_guest  stop    twinkle
frog     hope      mary         ring       summer  wall
```

The output might be divided into fewer or more columns depending on the width of your window.

If you add the "`-l`" (lower-case L) option, the output will instead list the files in your directory in an expanded format, with one file per line and information about each file's permissions, ownership, group, size, and last modification date:

```
-------$ ls -l /cas/course/cisc220/poems
total 76
-rw-r--r-- 1 lambm student  432 Sep  5  2016 bed
-rw-r--r-- 1 lambm student 2887 Sep  5  2016 birches
-rw-r--r-- 1 lambm cisfac   280 Sep  5  2016 frog
-rw-r--r-- 1 lambm student  925 Sep  5  2016 gentle
-rw-r--r-- 1 lambm student  447 Sep  5  2016 gustibus
-rw-r--r-- 1 lambm student  378 Sep  5  2016 hope
-rw-r--r-- 1 lambm cisfac   970 Sep  5  2016 jabberwocky
-rw-r--r-- 1 lambm student  269 Sep  5  2016 letter
-rw-r--r-- 1 lambm student  105 Sep  5  2016 mary
-rw-r--r-- 1 lambm student 1047 Sep  5  2016 michaelis
-rw-r--r-- 1 lambm student  797 Sep  5  2016 nov_guest
-rw-r--r-- 1 lambm student  639 Sep  5  2016 ring
-rw-r--r-- 1 lambm student  839 Sep  5  2016 road
-rw-r--r-- 1 lambm student  672 Sep  5  2016 stop
-rw-r--r-- 1 lambm student  655 Sep  5  2016 summer
-rw-r--r-- 1 lambm student  418 Sep  5  2016 tolls
-rw-r--r-- 1 lambm cisfac   183 Sep  5  2016 twinkle
-rw-r--r-- 1 lambm student 2015 Sep  5  2016 wall
-rw-r--r-- 1 lambm student 1464 Sep  5  2016 will
```

(The "total 76" at the start of the listing indicates the size of the directory – i.e. the size of the data structure describing the directory, excluding the sizes of the files themselves).

If you want output from `ls` to look different there are many, many options you can add to the command. Type

```
man ls
```

to see a "manual page" explaining all of the options available for the ls command. Using that information, figure out an `ls` command with options that will make the output different from the default in the following ways:

1. The output should be in a "long" format similar to what is shown for "`ls -l`" but it should not include the owner of the files ("lambm" for all files in this example)
2. The files should be listed in order of increasing size rather than in alphabetical order
3. Each file name should have quotes around it.

Now put that `ls` command into a bash script called `newls.sh` that takes the name of a directory as its one and only argument and lists the contents of the directory in the format described above. Here is an example:

```
-------$ newls.sh /cas/course/cisc220/poems
total 76
-rw-r--r-- 1 student  105 Sep  5  2016 "mary"
-rw-r--r-- 1 cisfac   183 Sep  5  2016 "twinkle"
-rw-r--r-- 1 student  269 Sep  5  2016 "letter"
-rw-r--r-- 1 cisfac   280 Sep  5  2016 "frog"
-rw-r--r-- 1 student  378 Sep  5  2016 "hope"
-rw-r--r-- 1 student  418 Sep  5  2016 "tolls"
-rw-r--r-- 1 student  432 Sep  5  2016 "bed"
-rw-r--r-- 1 student  447 Sep  5  2016 "gustibus"
-rw-r--r-- 1 student  639 Sep  5  2016 "ring"
-rw-r--r-- 1 student  655 Sep  5  2016 "summer"
-rw-r--r-- 1 student  672 Sep  5  2016 "stop"
-rw-r--r-- 1 student  797 Sep  5  2016 "nov_guest"
-rw-r--r-- 1 student  839 Sep  5  2016 "road"
-rw-r--r-- 1 student  925 Sep  5  2016 "gentle"
-rw-r--r-- 1 cisfac   970 Sep  5  2016 "jabberwocky"
-rw-r--r-- 1 student 1047 Sep  5  2016 "michaelis"
-rw-r--r-- 1 student 1464 Sep  5  2016 "will"
-rw-r--r-- 1 student 2015 Sep  5  2016 "wall"
-rw-r--r-- 1 student 2887 Sep  5  2016 "birches"
```

Script 2: A Script With Multiple Arguments. Write a script called `vacation.sh` that expects four arguments which describe a trip:

1. A year
2. A month
3. The name of a the place visited
4. The name of an activity you did in that place

Your script may assume that it will be called with exactly four arguments.

Your script must write a one-line description of the trip in the following format:

```
In <month>, <year> I visited <place> and enjoyed <activity>.
```

Here's an example showing some uses of my solution:

```
-------$ vacation.sh 2015 July Aruba snorkeling
In July, 2015 I visited Aruba and enjoyed snorkeling.
-------$ vacation.sh 2016 December Banff skiing
In December, 2016 I visited Banff and enjoyed skiing.
-------$ vacation.sh 2017 May London sightseeing.
In May, 2017 I visited London and enjoyed sightseeing.
```

Don't worry about checking the number of arguments or whether an argument makes sense as a month, year, destination or activity. This is simply an exercise in retrieving arguments and using them in output. So if someone calls your script with silly arguments your output may look silly too and that's OK. For example, with my solution:

```
-------$ vacation.sh chair zebra sandwich shoe
In zebra, chair I visited sandwich and enjoyed shoe.
```

Part 3: Filtering Names.

Let's arbitrarily define a "strange" file name as a name that contains both an x and a y, with the x coming before the y. The x and the y don't have to be next to each other in the file name; the only requirement is that the x has to come before the y. And if there are multiple x's and multiple y's in the file name, it's enough if one of the x's in the name comes before one of the y's.

Now write a script called `noStrange.sh` that takes the name of a directory as its one parameter. The script must delete every file in that directory which has a "strange" file name.

To avoid complications, you may assume that the directory exists and that there will be at least one file with a "strange" name in the directory. *You may also assume that the directory doesn't contain any subdirectories – so you don't have to worry about the possibility of needing to delete a subdirectory or files within a subdirectory. (This last sentence was added on Sept 19 in response to a student question.)*

You might be thinking that you will need to loop through the files in the directory and the letters in each file name. There are ways to do that in bash, but we haven't talked about them yet and you don't need them.

Hint: remember that the `rm` command can take multiple parameters and that you can use wildcard

sequences in the arguments to `rm`.

Here's an example using my solution. Suppose my current directory contains a sub-directory called `exampleDirectory`, containing several oddly-named files:

```
-------$ ls exampleDirectory
axbyc   baxxd      xopdy          xyabcde
aybcdx  cdxefgyyz  xxxxyyyyxxxx  yaybyyyycydxfxxxg
-------$ noStrange.sh exampleDirectory
-------$ ls exampleDirectory
aybcdx  baxxd  yaybyyyycydxfxxxg
```

Your `noStrange.sh` script should work whenever its parameter is a directory. It could be a sub-directory of the current directory (as in the example above) or it could be a full path name of a directory elsewhere, like `/bin/examples/someDirectory`.

What to hand in to the Assignment 1 Dropbox area: I will be creating this OnQ dropbox area several days before the due date. All you need to upload are your three scripts: `newls.sh`, `vacation.sh`, and `noStrange.sh`. Please use those exact names or we will give you an administrative penalty. We'll be using marking scripts to help us evaluate your scripts and if we have to stop and re-name scripts for lots of students it really slows down the process. If you are new to Queen's and/or OnQ and are unsure of the procedure for submitting assignments, please come to an office hour or a TA help session for a demonstration. Don't wait until the last minute to figure it out!

Marking Scheme: 4 points each for the 3 required scripts, for a total of 12 points. If you do not follow the assignment requirements -- especially if you deviate from them in a way that costs us time while marking -- you will lose points even if your scripts work correctly. So please read the directions carefully. This is part of programming.

Possible Penalties:

10% if you hand in during the late period (24 hours after the deadline)
10% if there are administrative issues that cause the graders extra time
If you are late AND have administrative issues you will get both penalties (20% off)