**CS 2204 Lab 7**

**your name here (please print):**

**your student ID number here:**

---

Create a subdirectory called `lab7` under your home directory. Perform any necessary work for this lab assignment in that directory.

---

This lab is a review of the use of quotes for 'escaping' purposes. We will use the command `echo` to illustrate the main ideas. First, do a `man echo` to determine (or recall) what `echo` does. Then, type the following suggested commands and understand why each of them work the way they do. Some parts below will require you to write answers (these are denoted by the point value assigned to them). Others are simply steps for which no answer is required from you.

1. Type

   ```
   echo hello world
   ```

   In this command, `echo` takes two arguments. The first argument is `hello` and the second argument is `world`. UNIX, by default uses space characters to separate the command name from its arguments, and to separate arguments from each other.

2. Type

   ```
   echo hello      world
   ```

   Here too `echo` takes two arguments but there is a lot of space between the first and second argument. Nevertheless, notice that UNIX produces the same behavior as the first command.

3. Type

   ```
   echo "hello world"
   ```

   In this command, `echo` takes only one argument. This single argument happens to have a space inside it. Normally, spaces are used to separate arguments (as in the previous two commands) but here we are interested in a single argument with a space inside it. To prevent UNIX from applying the default meaning of space as a delimiter, we use double quotes. When UNIX encounters a double quoted expression, it doesn't 'evaluate' this expression (by applying its usual rules), and instead passes the expression without judgment to the command (in this case, `echo`). It is important to note that, in this example, the single argument passed to `echo` is

   ```
   hello world
   ```

   i.e., without the double quotes.

4. (1 point) Looking at the output of commands in steps 1 and 3, you will see that there are no differences! So, what is the big deal? The reason is that, for this particular command, it doesn't really matter whether you pass 'hello world' as one argument or as two arguments separated by a space. `echo` by default produces a space between its outputs. For a different command, it can make a world of a difference. Name one such command (say, it is called XYZ) such that

   ```
   XYZ "hello world"
   XYZ hello world
   ```

   mean different things and cause (generally) different outputs.

5. (1 point) Similarly, name a command (say, XYZ) such that:

   ```
   XYZ *
   XYZ "*"
   ```

   mean different things and cause (generally) different outputs. Explain the difference. Notice that, in the first case, the asterisk is evaluated by UNIX (the shell, to be specific). In the second case, the asterisk is not evaluated and passed to the command `XYZ`.

6. To over-ride the default meaning of space, we used quotes. But what if we wanted to over-ride the default meaning of quotes? Answer: we use a backslash. For instance,

   ```
   echo Now type \"echo\"
   ```

   takes three arguments. The first is `Now`, the second is `type`, and the third is `"echo"` (with the quotes). We have just 'escaped the quote.'

7. This idea of using a backslash works in general. For instance, we can precede every space by a backslash, like so:

   ```
   echo hello\ \ world
   ```

   Note that this command has the same effect as

   ```
   echo "hello  world"
   ```

8. There is nothing special about the double quotes. The `echo` command allows the use of single quotes (') in exactly the same way. Repeat the above exercises using single quotes instead of double quotes and observe what happens.

9. We can even mix single quotes and double quotes, using one kind to escape the other kind. For instance, type

```
echo "'This text is surrounded by single quotes'"
echo '"This text is surrounded by single quotes"'
```

10. So what is the moral of the story? 'Escaping' means telling someone not to use the 'normal meaning' (whatever that might be). UNIX's normal meaning for space is a delimiter, so you have to escape the space if you want a literal space. One way to do it is to precede every space with a backslash. Another way is to enclose the string in double quotes (telling UNIX not to evaluate it). Similarly, UNIX's normal meaning of double quotes is to parcel arguments. If you want a literal space, you must escape the quote.

11. So, now that we know how to escape, should we use single quotes or double quotes? The answer depends on the particular UNIX command and the interaction between the command and the shell. We haven't covered this in detail yet, but single quotes are more powerful than double quotes. Stay tuned in future lectures.

12. (2 points) You are told that there is a file in your `lab7` directory called `*` (an asterisk). How will you print the contents of this file on the screen?

13. (2 points) What does the command `egrep CS2204* courses` do?

14. (2 points) How many arguments does the following command use?

```
echo "Hello "World""
```

15. (2 points) Write one or more `echo` commands to print the following onto the screen. Write your command(s) below.

```
A quote is ", a backslash is \, backtick is `.
A few spaces are   and dollar is $. $MY_VAR is 5.
```