CISC106 Summer 2009 Lab02

- Review the code examples from class.
- Some programs below are associated with a question. **Answer the questions** using comments below your code in the m-file.
- The office hours of the TAs and the instructor are on the class website. Visit us!

Problems

1. For this problem, you will create directories and move files around, and then show that you did so and draw a picture of what you did.

   (a) Make two new directories (some systems call them "folders") in your home directory on Strauss: lab01 and lab02. To do this, open an xterm on Strauss from the menu, then use the mkdir command as shown in your Unix text. Are there other ways to create directories? Yes, but this one will be on the exam. :)

   (b) Review mv, ls, and cd in your Unix text. Then use the mv command, as described below, to relocate the m-files and text files you have made in previous labs. For example, "circleArea.m" is in your home directory. You can verify this by typing in the shell:

   ```
   > ls
   ```

   to list all the files. Sometimes there are too many to easily see. In this case, you can use a "wildcard" character. Try this:

   ```
   > ls *.m
   ```

   This will show any files in this directory that end in ".m". Similarly, typing "ls c*" will show all files starting with the letter c.

   (c) Now that you have verified that "circleArea.m" is in your home directory, move it to your new lab01 directory:
   mv circleArea.m lab01/

   (d) Now use the cd command to change into the lab01 directory and verify that your file is there. Then go back up a directory level (use "cd ..") and finish relocating your lab files.

   (e) When you are done moving files, make a single diary file in Matlab: starting in your home directory, change into each of the directories you created and show the listing of the files there. Then end the diary. NOTE: never edit a diary file, not even to "fix" mistakes. A diary file is a record of your session, and modifying it is a violation of the Academic Honesty policy. If your diary session gets too messy, just start over.

   (f) After you print your diary file, draw a directory tree showing your home directory at the top, the lab directories in the next level down, and the lab files below that. Be tidy, and include this drawing when you hand in the paper version of your lab.

2. MATLAB provides a function for summing the elements of a vector, but the problem is a good demonstration of traversing (walking through the elements of) a vector with a loop.

   Enter your lab02 directory (how?). Copy the file sumInts.m from the class website's lab directory by using a Unix command (more practice!) in a shell (your xterm on Strauss is a shell):

```
cp /www/htdocs/CIS/106/tharvey/08F/labs/sumInts.m .
```

The dot at the end means "the directory I am in now" so that is where the file is being copied to. You could also say:

```
cp /www/htdocs/CIS/106/tharvey/08F/labs/sumInts.m ~/lab02/
```

See your Unix text for more information.

Now run the code by calling the function from the Matlab command line (Are you in the right directory? How do you find out?). Use assignment to store the value that the function returns to the interpreter.

3. Now copy the file sumIntsTest.m from the same directory and look at it. Note that it does not contain a function. This is a MATLAB script file. It contains MATLAB expressions that will be evaluated when you type the filename in the interpreter.

   A script M-file is not as nice as a function because you cannot specify parameters, and it shares variable names with the interpreter (how could that cause problems?).

   This script M-file is a "unit test" that is designed to run the examples from the sumInts function. Run the file and see if it works (it should).

   Now make three new unit test script files (you should always write the test file first! (why?)) for the functions sumOddInts, sumIntSquares, and sumOddIntSquares described below. Then write the functions themselves, and run your tests in a diary file.

   sumOddInts will use a for loop to sum all the odd integers between two parameters called "start" and "finish", e.g. `sumOddInts(5,9) -> 21`

   sumIntSquares will use a for loop to sum the squares of the integers from 1 to "finish".

   sumOddIntSquares will use a for loop to sum the odd squares of integers from "start" to "finish", i.e. 16 will be skipped but 25 would be added to the sum.

4. The function sumOddInts can be correctly written many ways. One common way is to test a number, and add it to the sum if it is odd; another is to start at an odd number, use an interval of 2, and thereby only generate numbers that are odd. Rewrite the function in a new file, sumOddInts2.m, in one of these two ways (but not the same way you did before) and modify your previous unit test file to test the new name.

   Type and demonstrate your function in a diary file.

Be sure that you have a printed copy of your four function M-files, four script M-files, diary files and a drawing. All must be stapled together, with your name and lab section on the top page. Be sure that you upload a copy of all the MATLAB files to Sakai. Then, click submit ONLY ONCE to send these to your Sakai and your TA.

On the first page of every printed copy for this course, your name, section, and TA's name must appear.