CISC106 Fall 2008 Lab06

- 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!
- **NOTE:** Every function comment section should contain, at a minimum, *three examples* of the function being called and the result of evaluating the call. Your test files must cover *at least* these exact examples (otherwise, why did you choose them?) and possibly more. Testing is important.
- **Every** M-file you write or modify must be demonstrated, either by running a script test file in a diary or by testing at the command line. Note that if you write function **foo.m** and test script **fooTest.m**, you can demonstrate both by running only fooTest.m (assuming fooTest works).
- New practice: Each problem below has a list of the files you are required to create. Use these names to receive full credit.

Problems

1. files: expt.m, exptTest.m

   Review the recursive function you wrote in lab 4, fact.m. Review your notes from class where we discussed what every recursive function needs to work correctly. Now write a recursive function to calculate the positive integer exponent of an integer, according to the following definition:

$$expt(base, exponent) = \begin{cases} base, \ if \ exponent \ is \ one; \\ base \ * \ expt(base, \ exponent - 1) \ otherwise. \end{cases} \quad (1)$$

   Hint: When the function calls itself, the base should stay the same every call, but the exponent should decrease (why?).

   Write and demonstrate a script test file for the expt function above.

2. files: distance.m

   See the file distanceTest.m in the lab directory. This is a script test file. Your first job is to write the "distance" function that will make this file work. The distance function computes the distance between two points in a cartesian plane (an x,y plane). See these web pages for more information:

```
http://math.about.com/library/bldistance.htm
http://purplemath.com/modules/distform.htm
http://en.wikipedia.org/wiki/Distance
```

   Run the test script on your completed function. DO NOT MODIFY the test script.

3. files: distanceTest2.m, checkTolerance.m

   In the file distanceTest.m, the same task is performed more than once: an actual value is compared to an expected value, the difference is compared to a tolerance, and a message is printed.

   Write a boolean function "checkTolerance" to handle this repetitive task. Your function will take three parameters: actual, expected, and tolerance. The output value for the function will be true if the test is passed, and false if the test fails.

   Rewrite the script file as distanceTest2.m, using your new function to replace some of the testing code wherever possible.

   It is not required, but you may also write a function to print outcome messages if you like - you'll find it useful, especially if you use it to compose meaningful messages.

4. file: whileInput.m

   Write a function that asks for a series of numbers, ending in the value -1. The function will then calculate the sum, mean and the count of numbers entered. Donot inlcude -1 in these calculations.

   Do not write a test script for this function, but demonstrate it in a diary file.

5. file: forInput.m

   Do the same as above but use "for" loop to do the job. For this, you need to know the exact count of input numbers to be typed by the user. Ask the user for this number and calculate sum and the mean.

   Do not write a test script for this function, but demonstrate it in a diary file.

6. files: usingfiles_while.m, (user specified) input_file, (user specified) output_file

   Write a function to ask the user to input the name of a file to read, name of a file to write. Then read (odd and even) numbers from the file which end with -1, count the odd numbers present in the file, their sum and mean. Write out all the odd numbers, their sum and mean into the file specified by the user.

   Do not write a test script for this function, but demonstrate it in a diary file.

7. file: usingfiles_for.m, (user specified) input_file, (user specified) output_file

   Write a function to ask the user to input the count of numbers, name of a file to read, name of a file to write. Read the (odd and even) numbers from the file, count the even numbers present, their sum and mean. Write out all the even numbers, their sum and mean into the file specified by the user.

   Do not write a test script for this function, but demonstrate it in a diary file.

If your TA requires a paper copy, be sure that you have a printed copy of your function M-files, script M-files, and diary files demonstrating your testing. 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 functions, scripts, and diary 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.