CISC106 Fall 2008 Lab08

- 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. We can change the size of a matrix we are using in Matlab. Show the following sequence of commands:

   ```
   x = []
   x(2) = 5
   x(2,3) = 80
   ```

   What is the size of x after each command? Use the size function to show this in a diary. Being able to ”grow” a matrix to fit our needs is a powerful tool, but not without cost. Write a function ”grow” that uses a loop to grow a vector one element at a time, from size one to size 10,000. Place a zero in each element. Write a second function ”preallocate” that uses the ”zeros” function to create a 1x10000 vector. Determine a number of runs of the ”preallocate” function that will give you measurable time. Do multiple timing runs of each function with a script. After the timing runs your script should show the size of the vector created by each function. Plot your data, showing each point (of course!). Post your labeled plot on your website. The plot title should include your name (see 2.11 and 3.5 in Chapman).

2. Write two functions. Write a function that uses a for loop and fprintf to print a row of n asterisks, where n is a parameter. Call the function from a separate ”wrapper” function[1] that asks for user input, then uses the input to make a line of asterisks that size. For example:

   ```
   >> asterisks()
   Please enter the size of the line: 5
   *****
   >>
   ```

[1] A wrapper is a function that wraps around another function to add utility. In this case we have a nice asterisk function that takes a parameter, and we write another function to handle I/O separately so we don’t have to modify the asterisk function.

Now change the wrapper function so that the user input and the call to the function are inside a while loop, and continue asking for length and printing asterisk rows until the user enters a negative number. (Hint: get the input before the loop starts; call the function at the top of the loop, and then get input again before the end of the loop).

```
>> asterisks()
Please enter the size of the line, or a negative number to stop: 6
******
Please enter the size of the line, or a negative number to stop: 13
*************
Please enter the size of the line, or a negative number to stop: -1
>>
```

Place both functions in a single M-file, with the wrapper function at the top. Be sure both functions have their own ”end”. Both functions must also have their own comments.

3. Copy the program from problem 2, and replace the function that prints asterisks with a for loop with a function that prints asterisks recursively. What will be a good base case? What are the traits of a base case?

4. Now use one of the asterisk functions you wrote to make squares. Write a new function that asks the user for the size of the square. If the size of the square is to be four, it will call the asterisk line function[2] four times:

```
What size would you like your square? 4
****
****
****
****
What size would you like your square? -1
Goodbye!
```

5. Write a Matlab function to determine the frequency with which an element appears in an array, e.g. in the example below, if the element in question was “8” your program should report that “8” appears four times in the test data. Accept the element to search as a parameter from the user.

```
A = [1 3 8 5 6 8 2 4 3 8 1 0 9 9 5 8 11 13];
```

6. In page 220 of your textbook, the test_ssort.m program uses ”for” loop to read the data. Convert this program to use ”while” loop, where user enters ”return” to end the data entry; use ”isempty” function to check for end of data.

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.

[2]Note: because you used a wrapper for I/O before, your asterisk function can be used here without changing it. Hoorah for wrappers!

Be sure that you upload a copy of all the MATLAB function, script, plots, 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.**