CISC106 Summer 2009 Lab07

- 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. These examples must include boundary conditions (as discussed in class). 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).
- **You can now work in pairs!** Please submit individually each problem, but you may work on the assignment with your partner. Please include the name of your partner with your submission.

Problems

1. Create a struct for a building that has properties for name, age, floors, and square footage. Show how to change the value of the name of the building. Create a second struct for a building and then assign the first struct to be equal to the second struct.

   Explain what you see when you display the values for both structs.

   Now change the name of the second struct. Display both structs again and explain what you see.

2. Create an array of building structs with different sizes for square footage. Create a function called **sizeLessThan** that compares two building structs and returns true if the first building is less than the size of the second building.

```
>> building1.squareFeet = 5;
>> building2.squareFeet = 10;
>> sizeLessThan(building1, building2)
ans =
   true
```

   Create a vectorized version of **sizeLessThan** called **vectorSizeLessThan** that compares a vector (one-dimensional array of buildings) to a second building and returns a logical array indicating whether each building in the vector is less than the given building. Hints: use your **sizeLessThan** function; also you can pre-create a logical array using false(1,n):

```
>> false(1,2)
ans =
   0    0
```

   Example:

```
>> building1.squareFeet = 5;
>> building2.squareFeet = 10;
>> vectorSizeLessThan([building1 building2], building2)
ans =
   0    1
```

3. Upgrade your buildings to class definitions. Create a

   `classdef`

   for building and give it the same properties as before and add a constructor function. Create a vector of buildings using the new class constructor. Try calling your **sizeLessThan** functions using these new buildings. Does this work? Explain why or why not.

4. Now create a parking lot class definition that only has two properties, spaces and squareFeetPerSpace. Show an example of creating a parking lot.

   Create a function called **getSquareFeet** that calculates the total square feet for a given parking lot:

   ```
   >> parkLot.squareFeetPerSpace = 100;
   >> parkLot.spaces = 50;
   >> getSquareFeet(parkLot)
   ans =
      5000
   ```

   Demonstrate your **getSquareFeet** working for parking lots.

5. Modify your **getSquareFeet** function to work for both parking lot classes and for buildings using the

   `isa`

   function in MATLAB. The

   `isa`

   function takes an object and a classname as arguments:

   `isa(building1, 'building') == true`

   Also create new **sizeLessThan** functions that use the **getSquareFeet** function. You should now be able to compare buildings directly to parking lots.

6. Graphic user interfaces in MATLAB. Load the example gui.m from the class examples directory. What do you notice about the functions and where they are declared?

   Add a File menu to the simple gui with two menu items, Load and Save; you should use the **uimenu** function in Matlab (do a google search for "matlab uimenu" for a usage reference). The Load item should call a function that loads the value of x from a file. The Save item should call a function that saves the value of x to a file. You may use a hard-coded file name for this exercise.

If your TA requires a paper copy, be sure that you have a printed copy of your function M-files, script M-files,image 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 function, script, imasge, 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.