Midterm Exam

Computer Programming for Engineering and Science
Held on 5th August, 2013 (CSCE 155M, Summer 2013)

Name :
Course No : **CSCE155M**

Instructions:

1. This is open book, open note, but not open neighbor.
2. If you have a question about the meaning of an exercise, ask! Getting things wrong because of misunderstandings can be aggravating for me as well as you.
3. If answers do not fit in the space provided, use the back of a sheet and very carefully indicate and label this so I don't miss it in grading.
4. You may take the entire period.

1. (10 points) Keeping in mind that a logical array is utilized quite differently from a numeric array, consider the follow interaction:

```
>> x = [3 5 1 7 6 2 4]
x =
     2     1     5     4     6     7     3
>> y = x > 3
y =
     0     0     1     1     1     1     0
>> z = [2 1 7 4 3 5 6]
z =
     2     1     7     4     3     5     6
```

   (a) What is the result of entering the following three expressions?

```
>> x(y)

>> x(z)

>> x(x)
```

2. (10 points) Consider the following code:

```
function f = fibo(n)
   if n <= 2
      f = 1;
   else
      f = fibo(n-2) + fibo(n-1);
   end
end
```

   (a) If this function is called with n being 6, how many times is *fibo(3)* called (recursively)?

   (b) What is the resulting value when n is 6?

   (c) Does the computer remember previous values for n that are restored as recursion is exited, or is there more than one variable named n that happens to have different values? Explain.

3. (10 points) Consider the following code.

```
function list = insert(e, list)
   small = list < e;
   big = list > e;
   list = [list(small), e, list(big)];
end
```

   (a) What happens with *l = insert(6, [2 3 5 8 9])* (sorted array)?

   (b) What happens with *l = insert(6, [1 5 6 6 6 8 11])* (element already there)?

   (c) What happens with *l = insert(6, [8 2 7 4 9 1 3])* (unsorted)?

4. (10 points) Consider the following code.

```
count = 0;
for a = 1:n
   for b = 1:n
      count = count + 1;
   end
   for b = 1:n^2
      count = count + 1;
   end
   for c = 1:n
      for d = 1:n
         for e = 1:log2(n)
            count = count+1;
         end
         for f = 1:n
            count = count+1;
         end
      end
   end
end
```

   (a) Give a formula for the final value to show how it can be calculated.

   (b) What is the big O complexity?

5. (10 points) Consider the following logic expressions.

   (a) Check the expressions in which short circuiting is enabled.

   (b) Circle the expressions in which short circuiting actually occurs.

   (c) Evaluate each expression.

   (d) Where does a side effect of a function take place, and what happens?

   Here are the expressions:

   (a) $25 > 357||false$

   (b) $-5 == 8||21 > 9$

   (c) $45 < 24|fprintf('howdy')$

   (d) $45 < 245\&fprintf'howdy'))$

   (e) $45 < 245\&\&fprintf('howdy')$

   (f) $true||2 < 71$

   (g) $2 - 16 < 6\&5 > 2$

6. (10 points) Write a piece of code that determines if *weather* is warm and wet, cool and dry, hot and dry, cold and wet, etc. assuming logical variable *precip* and numeric variable *degrees* with hot being over 90, warm over 80, mild over 70, and cold 60 or under, and prints out an appropriate message for each. Do this in each of the following ways. Time is of the essence, so if you can express an answer without being comprehensive, go ahead.

   (a) Separate *if* statements with no compound tests

   (b) One nested *if* statement (no *elseif*'s are allowed)

   (c) An *if* with *elseif*'s and no compound tests

   (d) A *switch* statement (tricky! Create a new integer variable.)

7. (10 points) Consider the following code. The variable *deck* is a vector of numbers from 1 to 52 represents playing cards. The number and order do not matter.

```
1 function [card deck] = draw(deck, n)
2   if nargin == 1
3      n = randi([1 length(deck)], 1);
4   end
5   pick = deck(n);
6   suits = {'Clubs','Spades','Diamonds','Hearts'}
7   card.value = mod(pick, 13) + 1;
8   card.suit = suits{ceiling(pick/13)};
9   if nargout == 2
10     deck = deck(deck ~= pick);
11  end
12 end
```

   (a) What happens if the user does not indicate a particular card?

   (b) Is the card determined by the position in the deck or the value at a given position? How can you tell?

   (c) Why is 1 added to the expression in line 7?

   (d) Why is *suits* a cell array instead of a regular array?

   (e) What is happening in line 10?

   (f) Why are lines 9 and 11 included?

8. (10 points)

   Consider the following function. *cFun* is intended to be the handle of a function which compares two elements in some desired way and returns *true* if the elements are NOT in the desired order.

```
function s = flexSort(a, cFun)
...
   if cFun(a(ii), a(jj))
      t = a(ii);
      a(ii) = a(jj);
      a(jj) = t;
   end
...
end
```

   (a) What happens in *flexSort* if the pair of elements being tested are NOT in the desired order? (Why is t needed?)

   (b) Write a function *compCards* that compares two cards as defined by the previous exercise and returns *true* if the first suit precedes the second suit or (if the suits are the same) the first card value is smaller than the second.

   (c) Rewrite your function as an anonomous function. Hint: Avoid using *if* statements. Instead use ONLY the test itself, which may be compound.

9. (10 points) A figure created by h = *plot(x,y)* continues to exist until it is explicitely deleted. While it exists, it can be modified, either directly with commands setting labels etc. or by the *set* function which, paired with *get*, can modify any of a number of properties of the figure.

   (a) h is the name of a handle in the above command. What does the value of h represent?

   (b) Using *get*, how would you get the complete listing of properties of the figure and their values?

   (c) Using *set*, show how you could triple the thickness of the plot line. (The property you need is *LineWidth*.)

Question	Points	Score
name	10	
1	10	
2	10	
3	10	
4	10	
5	10	
6	10	
7	10	
8	10	
9	10	
Total:	100	