Final Exam

Computer Programming for Engineering and Science
Held on 15th of August, 2013 (CSCE 155M, Summer 2013)

Name :

Save : **YES** **NO**

Course No : **CSCE155M Matlab**

Instructions:

1. This is open book, open note, open computer, but not open neighbor. Please do not use email, texting, etc. during the exam.
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 you wish to have the exam saved so that you can retrieve it later, please indicate this. Exams so marked will be saved until at least the second week of classes in the fall.

1. (10 points) Consider the following two snippets of code. How will the first respond to the inputs **3 * sin(pi/2)** and **6 * 2**? How will the second respond to the input **%d + %d = %d**? (Show exactly what will be assigned and printed.) Explain.

```
y = eval(sprintf('%s + %s', input('1st val: ', 's'), input('2nd val: ', 's')))
```

```
f = input('Enter format: ', 's');
z = fprintf(f, 15, 52, 15+52)
```

2. (10 points) Consider the following functions. What would be the response to the command line **x = nifty(2+3,3) * 4**?

```
function a = nifty(b,c)
t = b + neato(bummer(b), bummer(c));
a = t * bummer(t);
end

function x = neato(y,z)
x = y + z;
end

function x = bummer(y)
x = y - 2;
end
```

3. (10 points) Consider the following functions. What would be the response to the call **foo(6)**? Note that there is both a returned value and side effects.

```
function x = foo(n)
disp(n)
if n >= 1
   x = n + bar(n-1);
else
   x = 1;
end

function x = bar(n)
disp(n)
if n >= 1
   x = n - foo(n-1);
else
   x = 1;
end
```

4. (10 points) Consider the following function.

   (a) What would be the response to the call **fun**?

   (b) Rewrite it (the function and default x) so that it evaluates postfix notation expressions that are formed as nested cell arrays.

```
function t = fun(x)
if nargin == 0
   x = {'+', {'+', 2, 3}, {'*', 4, {'+', 5, 1}}};
end

if iscell(x)
   switch x{1}
      case '+'
         t = fun(x{2}) + fun(x{3});
      case '*'
         t = fun(x{2}) * fun(x{3});
   end
else
   t = x;
end
```

5. (10 points) Consider the following code that checks to see if an array is sorted. Create an anonymous function for each of the following cases that responds with **true** if the first argument is bigger than the second. Finally show how to invoke **isSorted** to determine your choice of the requests that follow:

   (a) The elements are numbers.

   (b) The elements are struct points with x and y coordinates. Size is determined by the distance from the origin.

   (c) The elements are struct fractions with numerator n and denominator d.

```
function s = isSorted(a, cFun)
   s = true;
   for ii = 2:length(a)
      if ~cFun(a(ii-1), a(ii))
         s = false;
      end
   end
```

6. (10 points) Consider the following code segment. Fill in the **fprintf** so that a shuffled deck of cards is displayed as

```
 3 of Hearts
10 of Spades
12 of Spades
 (etc.)
```

```
s = {'Clubs','Diamonds','Hearts','Spades'};
for c = 1:52
   card(c).suit = s{ceil(c/13)};
   card(c).val = mod(c,13)+1;
end
x = 1:52;
for ii = 1:100
   c1 = randi([1, 52]);
   c2 = randi([1, 52]);
   t = x(c1);
   x(c1) = x(c2);
   x(c2) = t;
end
for c = 1:52
   fprintf(   What goes in here????   )
end
```

7. (10 points) Consider the following lines. Which one is the better choice to use in a program. Would any not work as intended (restricting n to between 0 and 10 and checking that *a(n)* is 2)? Assume a is an array. Explain.

```
safe = 0 < n < 10 && a(n) == 2;

safe = 0 < n && n < 10 && a(n) == 2;

safe = 0 < n < 10 & a(n) == 2;

safe = a < n & n < 10 & a(n) == 2;
```

8. (10 points) In a normal queue, new elements are added to the *tail* using the *enqueue* operation while old elements are removed from the *head* using the *dequeue* operation. In a *priority queue*, the *enqueue* operation inserts the new element in a priority order, for example so that the smallest elements will be dequeued first. Write Matlab functions to do the following with queues of numbers:

   (a) Add an element e to priority queue q with the smaller elements having higher priority.

   (b) Extract the head element from q and return it as e.

   (c) Modify your enqueue function so that the user can specify how to prioritize the queue. (Hint: Review flexSort, or isSorted from an earlier exercise.)

9. (10 points) In Event-Driven Programming, the old master control loops have vanished. But in a 4-hand game of cards, you still really want the players to go in order. Consider the following piece of a callback function. How would you ensure that the card being discarded comes from the hand that is next to (clockwise) from the previous hand? Assume that the hands are laid out on the screen so that a player can poke on a card from his own hand but should not poke on a card from the other players' hands. All cards are set up with the same call-back function, but the field *UserData* contains a number from 1 to 4 representing the hand it is in. Note the global variable *turn* that is set to a number from 1 to 4.

```
function call_back(source, ~)
   global turn
   h = get(   What goes in here?   );
   if    (What goes here?)
      (good!  assume code exists here to process the play)
      (now that the play is processed, should we update 'turn' to next player?)
   else
      (bad!  display appropriate message)
      (Does 'turn' need to be updated here?)
   end
   (Does 'turn' need to be updated here?)
end
```

10. (10 points) Consider the following code. Note that file IDs 1 and 2 are preassigned to be standard output (the screen) and standard error (also the screen, but prints in red) respectively. Explain what is happening with the code assuming **text.txt** is text from a novel that naturally contains quoted material.

```
fid = fopen('text.txt');
q = 1;
while ~feof(fid)
   t = fgetl(fid);
   while length(t) ~= 0
      [w t] = strtok(t, '"');
      fprintf(q, w)
      if length(t) ~= 0 && t(1) == '"'
         q = 3 - q;
      end
   end
   fprintf('\n')
end
```

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