Midterm Exam II

Sample

COMPUTER PROGRAMMING FOR ENGINEERING AND SCIENCE
Held on 4th August, 2010 (CSCE 150E, SUMMER 2010)

Name :
Course No : **CSCE150E**

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) Consider the following two snippets of code. How would each respond to the user entering *mod(3+5,3)* in response to the input request? Explain!

end result is the same!

```
x = input('Enter an expression: ')
```

evaluated

2

```
y = input('Enter an expression: ', 's')
eval(y)
```

y becomes 'mod(3+5,3)'

2

2. (10 points) Consider the following code. Explain what happens, assuming ASCII files *list1.dat* and *list2.dat* each contains a list of positive integers in increasing order, ending with -1.

```
fid1 = fopen('list1.dat');
fid2 = fopen('list2.dat');
fid3 = fopen('list3.dat', 'w');

t1 = str2num(fgetl(fid1));
t2 = str2num(fgetl(fid2));

while t1 ~= -1 && t2 ~= -1
   if t1 < t2
      fprintf(fid3, '%d\n', t1)
      t1 = str2num(fgetl(fid1));
   else
      fprintf(fid3, '%d\n', t2)
      t2 = str2num(fgetl(fid2));
   end
end
while t1 ~= -1
   fprintf(fid3, '%d\n', t1)
   t1 = str2num(fgetl(fid1));
end
while t2 ~= -1
   fprintf(fid3, '%d\n', t2)
   t2 = str2num(fgetl(fid2));
end

fclose(fid1)
fclose(fid2)
fclose(fid3)
```

list1 and list2 are read in one number at a time, then "merged" to generate list3 which is fully sorted, but does not end with -1.

3. (10 points) What does the following code print when it is invoked from the command line with *funone*. Be reasonably careful with the spacing. Note that *blanks* is a built-in function which generates a string with the specified number of spaces.

```
function funone()
  n = 1;
  indent(n, 'Come in, if you dare!');
  funtwo(n*2);
  indent(n, 'Now you are in all the way!');
  funthree(n*2);
  indent(n, 'Good going, you got out!');
end

function funtwo(n)
  indent(n, 'intwo');
  funthree(n*2);
  indent(n, 'outtwo');
end

function funthree(n)
  indent(n, 'inthree')
end

function funthree(n)
  indent(n, 'inthree');
  if n <= 8
    fprintf('reCurses! Foiled again!')
    funtwo(n*2);
  end
  indent(n, 'outthree');
end
```

function indent(n)

(from previous test)

→1 come in
→2 in two
→4 in three
reCurses
→8 in two
→16 in three
→16 out three
→8 out two
→4 out three
→2 out two
→1 Now you are
→2 in three
reCurses
→4 in two
→8 in three
reCurses
→16 in three
→32 in two
→32 out two
→16 out two
→8 out three
→4 out two
→2 out three
→1 Good going

4. (10 points) Consider the following code, all in one file. Which function(s) is a candidate to be made into a nested function? Rewrite it to make it so in an optimal fashion.

both neato + bummer

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

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

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

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

function x = neato
x = b+c;
end

function x = bummer
x = t^2;
end

end
```

5. (10 points)

   (a) Describe the structure of *cella* after the first line (below) is entered.

   (b) Describe the structure of *cella* after the second line is entered. (You may simply rewrite the first line to incorporate the changes.)

   (c) Describe the structure of *cella* if the second line is performed five times in a row. (Perhaps sketch what it would look like?)

   (d) In the first case, how could one access the *bad* message?

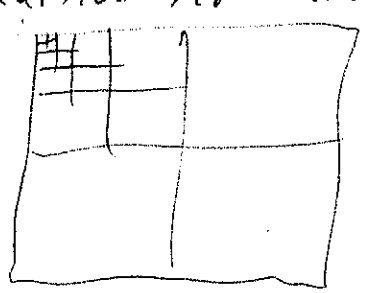
   (e) In the first case, how could one access the *[3 4]* of the array?

```
cella = {3+2i, 'hello'; [1 2; 3 4], {'good', 'bad'}};
cella{1,1} = cella;
```

a) 2 x 2 cell array

b) 1,1 position of cella (the 3+2i) is replaced with ~~copy~~ of original cella

c) recursive structure



d) cella{2,2}{2}

e) cella{2,1}(2,1:2) or:

Note: Don't take too much time trying to work out the formulas for colinearity and perimeter in the next two exercises. Do what you can, as that component will not cost you much if you do not get it. Do try to formulate the beginning and end of the functions.

6. (10 points)

   (a) Change the z coordinate of the second point of *triangle* to *27*.

   (b) Print the coordinates of the first point of *triangle*.

   (c) Write a function *isTriangle* to determine if the three points of a triangle are colinear (thus not forming a triangle).

```
point3D(3) = struct('x', 1, 'y', 3, 'z', 5);
point3D(2) = struct('x', 7, 'y', 9, 'z', 13);
point3D(1) = struct('x', 3, 'y', 5, 'z', 8);
triangle = struct('color', 'red', 'coordinates', point3D);
```

a) triangle.coordinates(2).z = 27

b) fprintf('%d %d %d\n', triangle.coordinates(1))

c) function s = slope(p1, p2)
s = (p1.y - p2.y) / (p1.x - p2.x)

function [xy yz zx] = slopes(p1, p2)
xy = (p1.x - p2.x) / (p1.y - p2.y)
yz = (p1.y - p2.y) / (p1.z - p2.z)
zx = (p1.z - p2.z) / p1.x - p2.x)

function x = isTriangle(t)
x = all(slopes(t.coordinates(1), t.coordinates(2)) ==
slopes(t.coordinates(2), t.coordinates(3)) &&
all(" (1) " (3)
" (2) " (3)) ;

end

7. Suppose a function to sort an array is written that permits the user to indicate how two elements are to be compared. For example, if the array is numbers, it might be with the function *gt* or *lt* (note that *gt(3,2)* is really the same as $3 > 2$.) Sorting an array of structures may require something more sophisticated, such as by the name field or by the age field. The array to be sorted is the first argument and how it is to be sorted is the second argument. The function definition begins as shown below. Assume ii < jj

   (a) Give the command to sort the numeric array *list* in increasing order.

   (b) Give the command to sort an array of triangles by increasing perimeter lengths (see previous exercise). named triangles

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

a) flexSort(list, @gt)

b)
```
function s = side (p1, p2)
    s = sqrt( (p1.x - p2.x)^2 + (p1.y - p2.y)^2 + (p1.z - p2.z)^2);
end

function p = perim (t)
    p = side (t.coordinates(1), t.coordinates(2)) + ...
        side (  "       (2),      "       (3)) + ...
        side (  "       (3),      "       (1));
end

flexSort (triangles, @isBiggerTriangle)

function x = isBiggerTriangle (t1, t2)
    x = perim (t1) > perim (t2)
end
```

8. (10 points) The function $max([52, 36, 17, 99, 5])$ returns two values: first, the value of the largest element (in this case it is 99), and second, the position of the largest element (in this case it is 4). Recall that arrays can be torn apart and rebuilt easily with Matlab. Recall that the opening line of a function that returns two results would look something like this: $function[x, y] = funny(a)$. Both x and y would presumably be assgned values somewhere in the function.

   Let array a consist of 52, 36, 17, 99, and 5.

   (a) What is the result of $b = a([1 : 2, 4 : end])$?

   52 36 99 5

   (b) What is the result of $b = [b, a(3)]$ (using the result of the previous part)?

   52 36 99 5 17

   (c) Write a function named *extract* containing one argument (an array) that extracts the largest element and returns both the extracted element and the remaining array (as two separate results).

```
function [big, rest] = extract(a)
[big p] = max(a);
rest = a([1:p-1, p+1:end])
end
```

   (d) What would be a better name for the following function?

```
function y = whatdoido(x)
y = [];
for ii = 1:length(x)
    [m x] = extract(x);
    y = [m y];
end
```

Increasing Sort

9. (10 points) The way you typically see an *fprintf* is with an opening string literal that represents the format for the following items. Note that in many instances you can use 3.14159 (a numeric literal) or pi (a numeric variable, though in this case it is predefined) interchangably. This is also true of string literals (in quotes) and string variables.

   (a) What is printed by the code (below)?

   (b) Modify the code by adding an *sprintf* function that generates the format string for the *fprintf* line.

ft= sprintf('The number %d is divisible by %s', f{1+d2+d3}); fprintf(ft, ii)

   (c) Modify the *fprintf* line (from the original code) so that it doesn't look so weird (in other words, the first argument is the more familiar string literal.)

```
f = {'one','two','three','six'}
for ii = 1:10
   d2 = mod(ii,2) == 0);
   d3 = 2 * (mod(ii,3) == 0);
   fprintf(['The number %d is divisible by ' f{1+d2+d3}], ii);
end
```

a) The number 1 is divisible by one
" 2 " two
" 3 " three
" 4 " two
" 5 " one
" 6 " six
7 one
8 two
9 three
10 two

c) fprintf('The number %d is divisible by %s', ii, f{1+d2+d3})

10. (10 points) A figure created by a command such as $h = plot(x,y)$ continues to exist until it is explicitly deleted. While it exists, it can be modified in various ways, either through functions like *xlabel()* or directly. There are many such *properties* that are associated with the figure that can be accessed and modified.

    (a) What is a *handle*? Can you point to an example above?

    (b) What functions are used to retrieve and to modify properties directly?

    (c) Give an example that will change the Marker property to plusses.

    (d) Give a couple ways that figures can be deleted.

a) reference to an object, h is an example

b) get and set

c) set(h, 'Marker', '+')

d) clf and just click on ☒

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