Final Exam

COMPUTER PROGRAMMING FOR ENGINEERING AND SCIENCE
Held on 12th of August, 2010 (CSCE 150E, SUMMER 2010)

Name :

Save : **YES NO**

Course No : **CSCE150E 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. There are 130 points in this exam. You are responsible for doing all the exercises. However, look at the score card on the last page. You will notice a column labeled *Revised* with a total of 100 points. You may revise downward the number of points of any or all exercises. For example, you could reduce the points for exercises 8, 9, 10, and 11 down to 5 each, and drop exercise 13 entirely. That would reduce the points by 30 which drops the total to 100. If you do not indicate any revisions, or if your revisions do not result in a total of 100, the original points will be used and your total scaled by 10/13.
4. 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 the second week of classes in the fall.

1. (10 points) Consider the following two snippets of code. Each responds the same to the user entering an expression such as *mod(3+5,3)* in response to the input request. Explain why this is the case!

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

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

2. (10 points) Consider the following code. Assuming ASCII files *list1.dat* and *list2.dat* each contains a list of positive integers in increasing order, ending with -1. It *merges* the two files into a single sorted file. Show how to modify the code so that the -1 values are not needed for determining that the end of the files have been reached. Note: The trickiest aspect is detecting and properly handling the case in which at least one of the files being read is empty.

```
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)
```

3. (10 points) Consider the following code, all in one file.

   (a) Rewrite the code to make the subfunctions nested, in an optimal fashion.

   (b) Rewrite the code to replace the subfunctions with anonymous functions.

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

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

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

4. (10 points)

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

   (b) How many times will Matlab let you repeat the second line?

   (c) After doing both lines, how could one access the *bad* message in the original cell array?

   (d) After doing both lines, how could one access the *[3 4]* in the nested cell array?

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

5. (10 points) 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.

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

   (b) Give the commands to sort the numeric array *list* in increasing order by the right-most digit, ignoring the other digits. Hint - the function *mod* might be useful.

   (c) Sort an array of points where *point* is a struct with fields x and y. Order by increasing distance from the origen. Hint: First write a function to calculate the distance of a point from the origen. Then write a function that compares distances of two points from the origen.

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

6. (10 points) Assume students identified by NU ID numbers are receiving numeric grades for their final exams. (Make up your own examples for the first part.)

   (a) Form the NU ID numbers into an array. Form the grades into a array. Sort the grades in descending order. Use the result to sort the ID numbers in descending order by grade.

   (b) While quicksort is generally very quick, it can become very slow when the array being sorted is nearly in order already. Suggest a command that should decrease the likelihood that the array of grades is already nearly sorted.

7. (10 points) A 2-D array named *pic* contains numbers in the range of 0 to 10. These represent brightness (10 being brightest) at points in a photo taken of the gulf off Louisiana. The oil is highly reflective, so brightness of oil covered areas is 8 or above. The following steps illustrate a simplified scenario that simulates the movement of the oil. In reality, wind would be a vector field, water currents would be another vector field, rate of spread would depend on a number of factors, etc., but all of these are easily handled by Matlab, and I hope suggest to you the importance of computation. Oh, try to avoid using loops!

   (a) Give a command(s) that will yield an array containing *true* for those locations likely covered with oil.

   (b) Give a command(s) that will calculate the area (number of elements that are true) of the oil covered water. (You may use the results of part (a) here and in the following part.)

   (c) Give a command(s) that will report how far north the oil has spread (the row with the smallest index containing a *true*).

   (d) Assuming oil spreads outward in all directions at the same speed, give a command(s) that show the extent of the oil after one unit of time (the time it takes to spread from one sampling point to the next).

   (e) Wind also affects the movement of the oil. Give the command(s) that take the result of the previous part, and show the affect if the wind blows this mess one unit to the east (right).

8. (10 points) Consider the following function definition. Now picture yourself at the command window, needing to use only that function to calculate the polynomial $6x^2 - 3xy - 7y^2$.

   (a) What do you need to type to do this using just one command?

   (b) Repeat, assuming the arguments to mOp are in the order *a*, *op*, *b*.

   (c) Repeat, assuming the arguments to mOp are in the order *op*, *a*, and *b*.

   (d) Using the answers above as a guide, express the polynomial using prefix, infix, and postfix notation, respectively.

```
function x = mOp(a, b, op)
switch op
   case '+'
      x = a + b;
   case '-'
      x = a - b;
   case '*'
      x = a * b;
   case '/'
      x = a / b;
   case '^'
      x = a ^ b;
end
```

9. (10 points) Consider the following code, which is very similar to what you have seen!

   (a) Cross off any redundant (unneeded) portions of the code. (The resulting code should always yield the same results.)

   (b) Using division by 10 and a correct choice of *ceil* or *floor*, rewrite the code to consist of a *switch* structure, requiring only a single test value for each non-F *case*. Note: Handling a grade of 0 is especially tricky, so there is plenty of partial credit if that is the only part you miss.

```
grade = input('Enter the grade from 0 to 100: ');
if grade > 90 && grade <= 100
    disp('Nice A')
elseif grade > 80 && grade <= 90
    disp('Not a bad B')
elseif grade <= 50 && grade >= 0
    disp('Sorry - you blew it')
elseif grade < 0 || grade > 100
    disp('Invalid input')
elseif grade > 60 && grade <= 70
    disp('Discouraging D')
elseif grade > 50 && grade <= 60
    disp('An E for effort is all you get')
elseif grade > 70 && grade <= 80
    disp('Average C')
end
```

10. (10 points) Consider the following code.

```
count = 0;
for a = 1:n
   for b = 1:n
      for c = 1:n
         count = count + 1;
      end
      for d = 1:n
         count = count + 1;
      end
   end
   for e = 1:n
      count = count + 1;
   end
end
```

(a) Give an algebraic formula using n for the final value of *count*.

(b) We say that the c and d loops are *triply* nested because each becomes the third active loop. What is the relationship between the largest exponent of n in the formula requested above and the maximum level of loop nesting?

(c) What is the (approximate) algebraic formula for the final value of *count* if the c loop is replaced with the following:

```
for c = 1:n
   x = n;
   while x >= 1
      count = count + 1;
      x = x/2;
   end
end
```

11. (10 points) Answer the following.

    (a) What is the basic idea of event driven programming?

    (b) What is the basic idea of GUI?

    (c) Does it make sense to have event driven programming without a GUI? Explain.

    (d) What is the basic idea of a call back function?

    (e) What is a handle, and how does a programmer work with it?

    (f) What functions are used to retrieve and modify properties of a GUI component directly? Give examples of both!

    (g) What is a uicontrol?

12. (10 points) What do each of the following functions return? Describe any side affects.

    (a) fprintf

    (b) input

    (c) sprintf

    (d) max

    (e) uicontrol

    (f) plot

13. (10 points) Write recursive functions to do the following:

    (a) Called as goUp(10), it counts from 1 to 10.

    (b) Called as doDown(10), it counts from 10 to 1.

    (c) Called as overHill(10), it counts from 1 to 10 and back down to 1.

    (d) Called as overDale(10), it counts from 10 to 1 and back up to 10.

Question	Points	Revised	Score
1	10		
2	10		
3	10		
4	10		
5	10		
6	10		
7	10		
8	10		
9	10		
10	10		
11	10		
12	10		
13	10		
Total:	130	100	