Midterm Exam II

COMPUTER PROGRAMMING FOR ENGINEERING AND SCIENCE
Held on 4th August, 2011 (CSCE 155N, SUMMER 2011)

Name : Key
Course No : **CSCE155N**

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) This is the same code as on the sample exam, with the assumption that the input files have numbers that are positive in increasing order, ending with -1. Modify the code so that it merges three files (rather than just two) into a new sorted file, and make the new file end with −1 also.

```
fid0 = fopen('list0.dat');
fid1 = fopen('list1.dat');
fid2 = fopen('list2.dat');
fid3 = fopen('list3.dat', 'w');
t0 = str2num(fgetl(fid0));
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
(repeat for t3)
fclose(fid1)
fclose(fid2)
fclose(fid3)
fclose(fid0)
```

```
while t0 ~= -1 && t1 ~= -1 && t2 ~= -1
   if t0 < t1 && t0 < t2
      (update t0)
   elseif t1 < t0 && t1 < t2
      (update t1)
   else
      (update t2)
   end
end
if t0 == -1
   (code for x)
elseif t1 == -1
   (code for x but using t0 + t2)
else
   (code for x but using t0 + t1)
end
```

2. (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?)

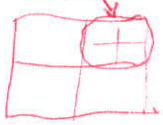
   (d) In the first case, how could one access the *bad* message?

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

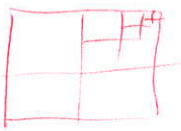
```
cella = {3+2i, 'hello'; [1 2; 3 4], {'good', 'bad'}};
cella{1,2} = cella;
```

a)

b)



c)



d) cella{2,2}{2}

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

3. (10 points)

   (a) Change the x coordinate of the third point of *triangle* to *27*.

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

   (c) Write a function *posTriangle* to determine if the three points of a triangle are all in the first quadrant (all coordinates are positive).

```
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(3).x = 27

b) fprintf('%d %d %d \n', triangle.coordinates(2).x, ...
triangle.coordinates(2).y, ...
triangle.coordinates(2).z)

c) function fq = posTriangle(t)

[a(1) b(1) c(1)] = triangle.coordinates.x;
[a(2) b(2) c(2)] = triangle.coordinates.y;
[a(3) b(3) c(3)] = triangle.coordinates.z;

d = [a b c]

fq = all(d > 0)

(10 points)

4. 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 command to sort an array of *entry* structures by increasing *grade* (which is one of the fields of *entry*).
   (c) Give the command to sort an array of *phone* structures by increasing dictionary order of *lastName* (which is one of the fields of *phone*).

named entries

number 0-100

named phones

alternately by length

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

assume jj > ii

could use directly

a) flexSort(list, @lt)

b) flexSort(entries, fn) fn = @(a,b) a.grade > b.grade

c) flexSort(phones, fn) fn = @(a,b) length(a.lastName) > length(b.lastName)

5. (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 triple the line width of a plot.

a) simple number used as a reference to an object
   example: h

b) get and set

c) set(h,'LineWidth', get(h,'LineWidth') * 2)

6. (10 points) Consider the following code, and invoking it with the line $frac(1, 2, 16)$.

   (a) What is printed?

   (b) What do the arguments (n, d, m) represent?

```
function frac(n, d, m)

if d <= m
   frac(n*2-1, d*2, m);
   fprintf('%d/%d ', n, d);
   frac(n*2+1, d*2, m);
end
```

a) 1/16 1/8 3/16 1/4 5/16 3/8 7/16 1/2 9/16 5/8 11/16 3/4 13/16 5/8 15/16

b) n - numerator
d - denominator
m - max divisions of demonimator

7. (10 points) Consider the analogies discussed in class.

   (a) Use the analogy of a computer with a human brain to come up with as many similarities and differences as you can.

   (b) Use the analogy of an office (with desk and file cabinets) with a computer to come up with as many similarities and differences as you can.

a) main memory — used for instructions & data
short term memory
calculator
control unit - instruction flow, I/O control
⋮

b) desk top = main memory
in/out boxes = open files
⋮

8. (20 points) Consider the code on the next page, which is all in one file.

   (a) As the function begins with x assigned as shown, what does *plist* print?

   (b) The variable x represents a *binary sorted tree.* Note that each cell array, including the nested ones, has a length of three. What do each of the three parts represent?

   (c) What does it mean for the first or third parts to be 0? to be a nested cell array?

   (d) Add comments to the code, specifically, explaining what each assignment statement does.

   (e) Assume the user types $a = stuff(12)$. What is printed by the function?

   (f) What is assigned to the variable a?

   (g) Now assume the user types $a = stuff(12, a)$. What is the effect of including the a as an argument?

   (h) ~~What~~ is ~~the~~ a new value assigned to a? Explain

a) 3 5 7 9 11 13 15

b) 1- left branch 2- value 3- right branch

c) 0- empty (no branch) cell array - the branch

d) (see next page)

e) 3 5 7 9 11 13 15
12 is larger than 7, looking right
9 11 13 15
12 is less than 13, looking left
9 11
12 is bigger than 11, looking right
not found, inserting it!

f) {{0,3,{0,5,0}},7,{{{0,9,0},11,{0,12,0}},13,{0,15,0}}}
replaces the 0

g) a is used as the tree instead of the default x

h) a stays the same since 12 is now found in it

```
function x = stuff(e, x)

if nargin == 1
   x = {{0,3,{0,5,0}},7,{{{0,9,0},11,0},13,{0,15,0}}};
end

plist(x)
fprintf('\n')

l = x{1};
v = x{2};
r = x{3};

if e == v
   fprintf('found the %d!\n', e);
   return
end

if e < v
   fprintf('%d is smaller than %d, looking left!\n', e, v);
   if iscell(l)
      x = {stuff(e, l), v, r};
   else
      fprintf('%d not found, inserting it!\n', e);
      x = {{0,e,0}, v, r};
   end
else
   fprintf('%d is larger than %d, looking right!\n', e, v);
   if iscell(r)
      x = {l, v, stuff(e, r)};
   else
      fprintf('%d not found, inserting it!\n', e);
      x = {l, v, {0,e,0}};
   end
end


function plist(x)

if iscell(x{1})
   plist(x{1})
end
fprintf('%d ', x{2})
if iscell(x{3})
   plist(x{3})
end
```

Question	Points	Score
1	10	
2	10	
3	10	
4	10	
5	10	
6	10	
7	10	
8	20	
Total:	90	