UNIVERSITY OF CALIFORNIA, COLLEGE OF ENGINEERING

E77N: INTRODUCTION TO COMPUTER PROGRAMMING FOR SCIENTISTS AND ENGINEERS

Final Exam—May 18, 2002
12:30–3:30 pm

TOTAL 100 POINTS

Notes:

1. Write your name below and on the top right corner of every page.
2. Check that you have all 14 pages of the test.
3. Please give all your answers only in the spaces provided.
4. You may NOT ask any questions during the exam.
5. Please no cell phones, calculators, or talking during the exam.

Your NAME: _____

Your STUDENT ID: _____

Your Signature: _____

For instructor use:

2	3	4	5	6	7	8	9	10	11	12	13	14	

Problem 1 (2 points) In the following table of temperature versus time, readings are missing at 2 and 5 hrs. Write three MATLAB statements that use linear interpolation to estimate the temperature at those two times.

Time, hrs	1	2	3	4	5	6	7
Temperature, ˚C	10	?	18	24	?	21	20

```
>> time = _____

>> temp = _____

>>_____
```

Problem 2 (6 points) The number of twists y required to break a certain rod is a function of the percentage x_1 and x_2 of each of two chemical components present in the rod. Write a MATLAB program that determines coefficients a_0, a_1, and a_2 by fitting the data given in the table below to equation $y = a_0 + a_1 \exp\left(x_1^2\right) + a_2\sqrt{x_2}$.

Number of twists y	40	51	65	72	38	46	53	67	31	39	48	56
Percentage of component 1 x_1	1	2	3	4	1	2	3	4	1	2	3	4
Percentage of component 2 x_2	1	1	1	1	2	2	2	2	3	3	3	3

Problem 3 (4 points) Given the following MATLAB function

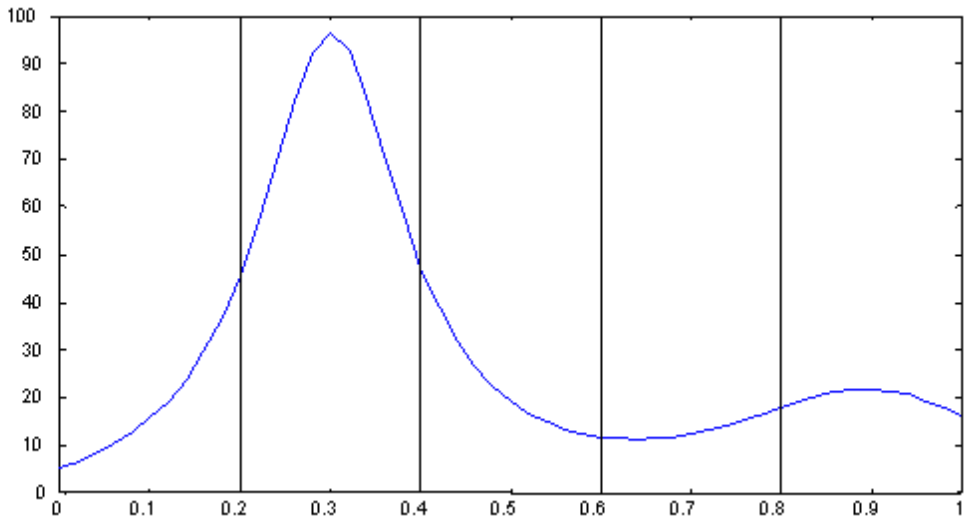
```
function v =velocity(t)
v = 2 * t + 1;
```

what will be the result from the following MATLAB commands

(a) `>> quad('velocity', 0, 1)` _____

(b) `>> diff(velocity([0:2]))` _____

Problem 4 (3 points) The curve below represents a function $y = f(x)$. Show graphically, on the same diagram, how the trapezoidal rule evaluates the integral $\int_0^1 f(x)\,dx$ with the step size of 0.2. Provide a brief explanation of what you showed on the graph.



Brief Explanation:

Problem 5 (4 points) Complete the Newton-Raphson root finding function:

```
function xnewt = newton(fhandle,_____,_____)
for j = 1:15
    [f,df] = ___________(_____,_____);
    dx = _________________;
    xnewt = _____________;
    if (_____(_____) < xtol)
        return
    end
end
```

Problem 6 (5 points) By mathematical induction prove $S_n = \sum_{k=1}^{n} k = \frac{n(n+1)}{2}$ for $n \geq 1$

1. What is *n* equal to for your base case? Does it check?

2. What is the recurrence relation you will need in your proof?

3. What will you assume is true in performing the proof?

4. What do you need to show is true?

5. Complete your proof by proving the statement you wrote down in step 4.

Problem 7 (8 points) Write a recursive MATLAB function to evaluate the value of *y* from the following equation where *x* is an array of length *n* containing non-zero elements.

$$y = x(n) + \cfrac{1}{x(n-1) + \cfrac{1}{x(n-2) + \cfrac{1}{\ldots + \cfrac{1}{x(2) + \cfrac{1}{x(1)}}}}}$$

Problem 8 (5 points) (a) Draw a binary search tree for the following array of numbers

```
[35, 64, 12, 45, 16, 85, 28]
```

such that *inorder traversal* of the tree visits the numbers in an ascending order. Use the first element as the root of the tree.

(b). For the binary search tree developed in part (a) above, show the data structure with three fields (key, left pointer, and right pointer) in the boxes given below.

array index	1	2	3	4	5	6	7
key	35	64	12	45	16	85	28
left							
right							

Problem 9 (3 points) Show, in the table below, the consecutive steps of the bubble sort for array [16 12 64 45] in an ascending order. Note that more columns might be given than required.

16								
12								
64								
45								

Problem 10 (2 points) For the code below, give a big-O running time bound in terms of N

```
for i = 1:N
   s = 0;
   for j = 1:N
      s = s + a(i,j) * b(j);
   end
   c(i) = s;
end
```

Mark your answer:

___ $t \sim O(N^0)$
___ $t \sim O(N^1)$
___ $t \sim O(N^2)$
___ $t \sim O(N^3)$

Problem 11 (4 points) MATLAB function `perms(1:N)` returns all possible permutations of array `[1:N]`. Running times for several values of N are given below

N	5	6	7	8
Running time (sec)	0.1	0.5	3.5	28.8

Write a MATLAB code that uses the data in the table above and determines the big-O scaling (i.e. the power of N) for this function.

Problem 12 (4 points) Class `fraction` is defined by functions `fraction.m` and `display.m`, both placed in directory `@fraction`; their listings are given below.

```
function r = fraction(a,b)
% r = fraction(numerator,denominator)
% constructor for class fraction
%
r.numer = a;
r.denom = b;
r = class(r,'fraction');
```

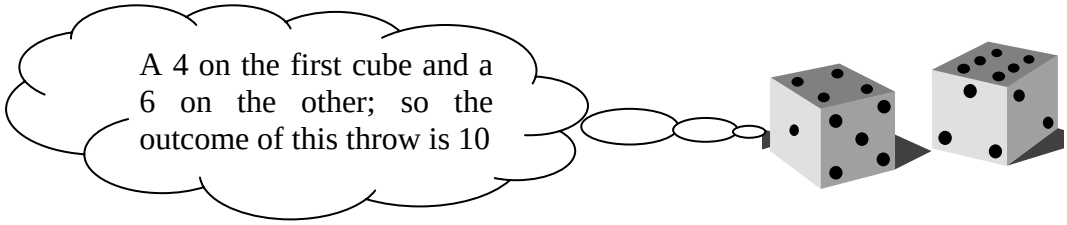
```
function display(r)
% displays object r of class fraction
%
if (r.numer < r.denom)
   disp([num2str(r.numer) '/' num2str(r.denom)])
else
   w = floor(r.numer/r.denom);
   f = rem(r.numer,r.denom);   % determines remainder of a/b
   if (f == 0)
      disp(w)
   else
      disp([num2str(w) ' + ' ...
            num2str(f) '/' num2str(r.denom)])
   end
end
```

Write the answers produced by the following MATLAB statements typed in the MATLAB command window:

a) `>> x = fraction(1,2)` _____

b) `>> x = fraction(7,5)` _____

Problem 13 (10 points) Write a simple MATLAB program to simulate the throw of two dice each with face values of 1 to 6. The program is to determine the number of times the outcome of [1, 2, 3, 4, ..., 11, 12] is observed after 100 throws.



As an example, if the outcome of 10 throws is 11, 3, 6, 9, 5, 6, 5, 3, 6, and 2, the number of times the outcome of [1,2,...12] observed is [0,1,2,0,2,3,0,0,1,0,1,0].

Problem 14 (10 points) Write the MATLAB code necessary to plot *y* versus time, from 0 to 100 seconds, by solving the following set of ODEs with the initial condition of zero for all the variables,

$$\ddot{y} - y\sin(x) = 0$$

$$2\dot{x} = 3y - x$$

Problem 15 (10 points) You are given a structure array `student` with two fields: `name` and `score`.

(a) Write a MATLAB code that prints students names (stored in the field `name`) according to their scores, from the highest to the lowest. (HELP: Given array `X`, `SORT(X)` returns an array with elements `X` sorted in an ascending order; `[Y,I] = SORT(X)` also returns an index matrix `I`).

(b) Write a MATLAB code that computes students ranks and store them in a new field, `ranking`, of the `student` structure array. The student with the highest score receives the first rank, next lower score receives a rank of 2, and so on. However, students with equal score receive equal rank.

Problem 16 (10 points) The following is a MATLAB function for inorder traversal of a binary search tree:

```
function v = inorder(index)
% The function takes an integer argument index, which is
% the index to the node of the global structure array bnode,
% and prints the tree inorder.
% The structure bnode has 3 fields: key, left, and right.
%
global bnode;
if (bnode(index).left ~= 0)
   inorder(bnode(index).left);
end
disp(bnode(index).key);
if (bnode(index).right ~= 0)
   inorder(bnode(index).right);
end
v = 'OK';
```

Modify the above function to return the number of leaves of a binary search tree. Assume that the call to the function is `inorder(1)`.

```
function v = inorder(index)

global bnode;
```

Problem 17 (10 points) The structure assignments below define a graph:

```
gnode(1).name = 'A';
gnode(1).neighbors = [2 3];
gnode(2).name = 'B';
gnode(2).neighbors = [1 4 5];
gnode(3).name = 'C';
gnode(3).neighbors = [1];
gnode(4).name = 'D';
gnode(4).neighbors = [2 5];
gnode(5).name = 'E';
gnode(5).neighbors = [2 4];

for i = 1:5
   gnode(i).flag = 0;
end
```

(a) Neatly draw the graph with the nodes labeled by letters.

(b) Write the adjacency matrix for the graph defined above.

(c). For the graph developed in part (a), list the nodes in the order that depth-first search would visit them, starting with the call `dfs(1).` The code for `dfs` is shown below:

```
function v = dfs(index)
%
% depth-first search of a graph.  The argument index is
% the node to start searching from.
% Each node visited is printed.
%

global gnode;

if (gnode(index).flag == 0) % Has this node been visited before?
   gnode(index).flag = 1;   % If not, set its flag
   disp(gnode(index).name); %   and print the name of the node

   for i = gnode(index).neigbors   % then visit its neighbors
      dfs(i);
   end

end
```