Name:_____

ID#:_____

Enrolled Discussion (circle one): M–10am M–11am M–7pm F–10am F–11am F–12pm

MAE10
Final Examination
Winter Quarter 2010

Instructions: You have 120 minutes to complete the exam. Notes on **two** sides of an 8.5''x11'' sheet of paper are allowed. Closed book. No calculators or electronic devices of any kind.

Section 1: Short answer. (2 points each)

(1.1) How many rows and columns are in the arrays w and z?

```
x = (-1:2:3);
y = (3:-2:-1);
w = [x, y];
z = [x; y];
```

(1.2) How many data lines will be plotted on the figure generated by the following code?

```
t =(0:1:10);
x =t.^2;
y =x.^2;
plot(t,x,y,x,t,t);
```

(1.3) What value(s) are stored in y?

```
array = [ 1, 3, 5, 6  ;  8, 9, 10, 11 ];
y = numel( array );
```

(1.4) What value(s) are stored in x and y?

```
array = [ 1, 3, 5, 6  ;  8, 9, 10, 11 ];
[ x , y ] = size( array );
```

(1.5) What value(s) are stored in y?

```
x = [ 1 , 3 , 4 , -2 , -1];
y = find( x >= 3 );
```

Section 2: Identify and briefly explain any and all ***errors*** that would prevent the code from executing in the following MATLAB programs. Warnings and "bad programming" are not considered errors – those would not prevent the code from running. If you believe there are no errors, write "OKAY". (3 points each)

(2.1)

```
x = ( 2 : -1.5 : -1 );
y = ( 6 : 2 : 12 );
w = x + y;
x(2) > x(3) & y(1) ~= y(2);
```

(2.2)

```
cheese = 'mouse';
goose = 'rat';
mouse = 'goose';
couscous = 'house';
moose = [ cheese(1:1) , goose(2:5) ] ;
fprintf('mouse is %s' , couscous)
```

(2.3)

```
x = [ 2:2:10 ; -1:1:3 ];
a = min( x(1,:) ) ;
b = max( x(2,:) ) ;
if( a >= b )
   disp(a)
else
   disp(b+a)
   if(x(2,2) ~= x(1,1) & x(1,1) > x(1,1))
        disp(b)
        if( a ~= b )
             disp(a*b)
        end
   end
end
```

(2.4)

```
x = [ 1, 5, 7, 8 ; 9, 1, -1, 2 ] ;
y = [ 2, 4, 6, 8, 10 ] ;
for k = 1:4
   w(k) = x(2,k) + y(k) ;
   for m = k:4
        h(m) = h(m) + w(k) ;
   end
end
fprintf( '%7.2f  %3i' , h )
```

(2.5)

```
y = [ 2, 4, 6, 8, 10 ] ;
k = 2;
while( y(k) <= 6 )
   disp( y(k) )
   k = k + 1;
end
```

(2.6)

```
animal = 'moose';
switch (animal)
   case{ 'MOOSE' , 'MoOsE' }
        disp(animal)
   case{ 'goose , horse' }
        disp( 'goose or horse' )
   case{ 'hoof' , 'antlers' }
        disp('moo')
end
```

(2.7)

```
t = (0:0.1:1);
x = t.^2
y = t - 1
z = t.^2
subplot(2,1,3)
plot(y,x)
subplot(2,1,2)
plot(t,t)
xlabel('The y-axis')
subplot(2,1,1)
plot(x,y)
grid on
```

For problems (2.8) through (2.10), I provide a function and the "main" program that calls the function, each stored in a different M-File. Assume both M-Files are in the same directory. Errors may exist in either or both programs.

(2.8)

In the "main" program:

```
a = 3.3;
b = 4.4;
[c , d] = banana(a , b)
fprintf('a and b are %5.2f and %5.2f', a, b)
```

In the m-file containing the function `banana`:

```
function [ result ] = banana(b , a)
a = b
b = a
result = b + a
end
```

(2.9)

In the "main" program:

```
a = 3.3;
b = 4.4;
answer = gorilla(a , b)
fprintf('answer is %7.2f' , answer)
fprintf('c and d are %7.2f %7.2f' , c , d)
```

In the m-file containing the function `gorilla`:

```
function [ result ] = gorilla(c , d)
c = 100;
d = 1000;
result = c + d;
end
```

Section 3: Write the ***exact*** output that will be produced by each of the following programs. Assume that there are no errors. Clearly distinguish answers from any scratch work and indicate which line of code each answer corresponds to. If you are given specific formatting instructions, you must **use underscores to indicate any and all blank spaces** (one underscore per blank space). In addition, carriage returns (newline) should be clear. (4 points each)

(3.1)

```
a = [ 1, 3, 5, 7 ];
b = [ 5, 7, 8, 9 ];
c = [ a ; b ]' ;
c = c(2:3 , 1:2);
fprintf( '%3i\n' , c )
```

(3.2)

```
a = [ 12, 2, -2, 1, 0 ];
b = (a(2)^a(5)+a(2)/a(3)-a(1))/a(2)+a(1);
c = a(5)-a(3)/a(2)-a(1)^a(5);
d = 2;
for i = 1:numel(a)
   d = d + a(i)^a( -2*a(3) );
end
fprintf( '%6.1f\n' , b, c, d)
```

(3.3)

```
a = ( 1 : 3 : 101 );
k = 1;
while(k<5)
   fprintf( '%7.2f%5.2f\n',a(1:k))
k = k + 2;
end
```

(3.4)

```
cheese = 'swiss';
a = [ 99, 98, 97, 96, 95 ];
switch  cheese
   case {'SWISS' ,  'cheese'}
        disp( cheese(2:5) )
        fprintf( '%5i' , a(3:5) )
   case {'cheddar' , 'swiss'}
        disp( cheese(1:4) )
        fprintf( '%3i' , a(1:3) )
   case {'swiss'}
        disp( cheese(1:5) )
        fprintf( '%3i' , a(1:5) )
   otherwise
        disp( cheese )
        fprintf( '%3i' , a(2:3) )
end
```

(3.5)

```
for col = 1:2
   for row = 1:3
        b(col,row) = col + row;
   end
end
fprintf( '%5i%4i\n' , b )
```

(3.6)

```
a = (1:4);
b = a*2;
c = b*2;
if( c(2) < a(4) | c(2) == a(4) )
   a = a + 1;
   if( c(1) <= b(2) & c(2) > a(4) )
        fprintf('Hello world\n')
        a = a + 1;
   elseif( c(1) > a(2) | c(3) > 9 )
        fprintf('Hola mundo\n')
        b = b - 1;
   else
        c = c*2;
   end
else
   b = b*2;
   if( c(1) == b(1) )
        fprintf('Hallo Welt\n')
        c = c - 1;
   elseif( c(4)/a(1) > 3 )
        fprintf('Howdy Mundo\n')
        a = a + 3;
   end
end
fprintf( '%2i%3i\n' , a , c )
```

For problems (3.7) through (3.10), I provide a function program and the "main" program that calls the function. Assume both programs are in the same directory.

(3.7)

In the "main" program:

```
a = (10:2:15);
b = (11:2:15);
fprintf( '%3i%3i\n' , a(2:3) )
fprintf( '%3i%3i\n' , b(1:2) )
[ b , a ] = func1( a , b );
fprintf( '%3i%3i\n' , a(2:3) )
fprintf( '%3i%3i\n' , b(1:2) )
```

In the m-file containing the function `func1`:

```
function [ a , b ] = func1( b , a )
temp = a;
a = b;
b = a;
end
```

(3.8)

In the "main" program:

```
a = (1:1:3);
b = (2:2:6);
table1 = [ a ; b ];
fprintf( '%2i%2i\n' , table1 )
[ x , y ] = func2( a , b );
table2 = [ x ; y ];
fprintf( '%2i%2i\n' , table2 )
```

In the m-file containing the function `func2`:

```
function [ a , b ] = func2 ( x , y )
for i = 1:2
   a(i) = y(i);
   b(i) = x(i+1);
end
end
```

(3.9)
In the "main" program:

```
a  = 4.5;
b  = 2.5;
c  = f4(b) / f3( a , b );
d  = f4(a) * f4(a);
fprintf('%6.1f' , c , d , f4(a) )
```

In the m-file containing the function `f3`:

```
function [ output ] = f3 ( x , y )
output = x + y;
output = output - 3;
return
output = output/2;
end
```

In the m-file containing the function `f4`:

```
function [ output ] = f4 ( x )
output = 1;
for i=1:2
   output = output + x;
end
end
```

(3.10)
In the "main" program:

```
a  = 2;
b  = 3;
c  = f5(a,b) + f6(b,a);
fprintf('%3i\n' , c)
```

In the m-file containing the function f5:

```
function [x] = f5(x,y)
a = x + y;
b = x - y;
x = f6(a,b) + f6(b,a);
fprintf('%3i%3i\n' , a , b)
end
```

In the m-file containing the function f6:

```
function [x] = f6(x,y)
a = x*y;
b = x + a;
x = b + a;
fprintf('%3i%3i\n' , a , b)
end
```

Section 4: Write a MATLAB program to solve each of the following problems. You do not have to write the output of the code.

(4.1) You are provided with a data file called "medal.txt" containing the Olympic medal count for 100 nations. (7 points)

In "`medal.txt`"

```
1     5     7
2     4     0
5     1     0
… <rows 4-98 not shown> …
2     9     0
1     1     11
```

Each row contains information about the number of gold, silver, and bronze medals won by each nation. The first column is the number of gold medals, the second column is the number of silver medals, and the third column is the number of bronze medals. For example, nation #2 won two gold medals, four silver medals, and zero bronze medals.

Create a program that will read the data from "medal.txt" and store it in an array. Then find out the following information and display the answers to the screen:

(1) Which nation (1-100) had the highest number of gold medals?
(2) Which nation (1-100) had the highest number of silver medals?
(3) Which nation (1-100) had the highest number of bronze medals?
(4) Which nation (1-100) had the highest total number of medals?

(4.2) The expressions for x, y, and z are as follows:

$$x = t^2 + 2$$
$$y = t - 1$$
$$z = x * y$$

Create a program that calculates x, y, and z from t = 0 seconds to t = 10 seconds in increments of 0.1 seconds. Plot the following information on three separate subplots. (4 points)

(1) In the upper right section of the plotting window, plot x versus t.
(2) In the center section of the plotting window, plot y versus x.
(3) In the center right section of the plotting window, plot z versus y.

x versus t		
	y versus x	z versus y

(4.3) A program calls a function named `switchrow`:

```
A = switchrow(A,M,N)
```

Create a function named `switchrow` that exchanges the first and last rows of a `MxN` array (called A). For example, if the array A is,

```
4 5 8 1
0 3 6 7
2 3 5 1
0 0 0 9
```

Then the function will create an array with the following elements,

```
0 0 0 9
0 3 6 7
2 3 5 1
4 5 8 1
```

Make your function general for any MxN array. You <u>cannot</u> use any of MATLAB's built-in functions for this problem except functions that determine the number of elements in an array. (5 points)

(4.4) A program calls a function named `remainder`:

```
remain = remainder(x,y)
```

Create a function named `remainder` that will calculate the remainder of the variable x divided by the variable y. You cannot use MATLAB's built-in function rem( ) for this problem. (4 points)

(4.5) A program calls a function named `sortscores`:

```
sortedscores = sortnums(scores)
```

Create a function named `sortnums` that takes as input a 1xN array called scores, which contains test scores from a class. This function returns as output a 1xN array called `sorted`, which contains the same scores sorted from lowest to highest. For example, if `scores` contains the values [84.5, 90.1, 56.0, 92.9, 77.0], then `sortedscores` should contain the values [56.0, 77.0, 84.5, 90.1, 92.9]. Make your function general for any size array. You may assume that all the scores are unique (there aren't duplicate numbers).

You cannot use any of MATLAB's built-in functions except those functions that determine the number of elements in an array. (7 points)