Instructions: You have until the end of the class period to complete the exam. Notes on both sides of an 8.5”x11” sheet of paper are allowed. Closed book. No calculators.

Section 1: Short Answer. (2 points each)

1.1 Name one positive and one negative aspect of cloud computing?

1.2 How many data lines will be plotted on the figure generated by the following code?
   
   ```plaintext
t = [-1:100];
x = t.^2; y = x.^2; w = y.^2; z = w.^2;
G = [ t ; x ; y ; w ; z ];
plot(t,G)
```

1.3 How many elements are in w?

```
y = [ 1:3:10 ; 2:3:11 ; 3:3:12 ];
w = y( 1:2:3 , end );
```

1.4 What value(s) are stored in t?

```
x = [1 2 3 4]; y = [4 3 2 1];
w = (x >= y);
t = find(w)
```

1.5 What fraction (0% to 100%) of the plotting window will be occupied by subplot(s)?

```
x = [1:10]; y = x.^2;
subplot(2,2,3)
plot(x,y)
```
(1.6) Convert the following binary (base-2) numbers to decimal (base-10) numbers. The decimal number system is what we use every day to count (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, ...).

0101011 1010101

Section 2: Identify all errors in the following segments of code, if any exist, and give an explanation of the error. If you believe that there are no errors write “No errors.” (2 points each)

(2.1)  
A = [1 2 3 4 5 6 7 8 9 10 15 16 17 18 19 20];
B = [1 3 5 7 9 0];
C = A(1:5) + B(2:6);
D = C + A(11:2:15);

(2.2)  
X = (100:200);
Y = (1000:2000);
tablexy = [X(3:end) , Y(4:end).^2 ];
tablexy = tablexy(1:2)

(2.3)  
X = [1:2:11 ];
Y = [1:3:10 ];
if( X > Y )
  disp(X)
  elseif( (X(1) > Y(2)) | (X(1) < Y(2)) )
    if( Y(3) > Y(2) )
      disp( X(4) )
    end
  elseif( ~(X(2) < X(3))
    disp( X(5) )
  else
    disp( X(4) > X(3) )
end
\begin{enumerate}
\item \[ A = [ 1:3 ]; \]
\[ B = [ -2:3 ]; \]
\text{for k = A}
\text{  i = 2;}
\text{  while( i<5 )}
\text{    C(k) = C(k) + A(k);}
\text{  i = i + 1;}
\text{end}
\text{end}
\text{fprintf( '%7i\n' , A,B,C )}
\item \[ Y = [34.5, 65.22, 89.09, -12.34]; \]
\[ X = [43.2, 76.5, -90, 0]; \]
\text{fprintf('8.4f\n' , X, Y)}
\text{fprintf('g g %e\n' , [X, Y])}
\text{fprintf('i g %9.1f\n' , [X ; Y])}
\item \[ A = '51152ABC'; \quad B = 'Y1111X'; \quad C = '15119012'; \]
\text{switch ( C(3:4))}
\text{  case{ A(1:3), A(1:2) }}
\text{    disp('hi')}
\text{  case{ B(2:3), B(3:4) }}
\text{    disp( 'bye' )}
\text{  case{ A(6:7), '11321' }}
\text{    disp('pickles')}
\text{end}
\item \[ X = [0:10:100 ]; \]
\[ Y = X.^3; \]
\[ T = Y .- X; \]
\text{subplot(4,2,5)}
\text{plot(X,X)}
\text{subplot(4,2,9)}
\text{plot(T,X)}
\text{title('My fun figure')}\quad \text{subplot(4,2,1)}
\text{plot(X,Y)}
\end{enumerate}
For problems (2.8) through (2.10), I provide a function program 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 both programs.

(2.8)
In the “main” program:
```matlab
X = [ 2:2:10 ];
Y = [ 3:3:1200 ];
[X,Y] = pizza(X, Y);
fprintf('%i', [X, Y])
```

In the m-file containing the function `pizza`:
```matlab
function [ W,T ] = pizza(W, T)
W = T';
endfunction
```

(2.9)
In the “main” program:
```matlab
X = [ 1:10 ];
Y = [ 2:11 ];
W = [ 3:12 ];
output = f1(X(2:4), Y(9:10), W(5:7));
fprintf('%7.2f', output)
```

In the m-file containing the function `f1`:
```matlab
function [ result ] = f1(A,B,C)
A = A + 2;
B = B + 2;
C = C - 2;
result = A + B - C;
endfunction
(2.10)

In the “main” program:

\[
\begin{align*}
X &= 3; \\
Y &= 4; \\
A &= 10; \\
B &= 20; \\
W &= f1(X,Y) + f2(A,f1(B,B)); \\
fprintf('W = %f', W)
\end{align*}
\]

In the program containing the function \( f1 \):

\[
\begin{align*}
function [ Y ] &= f1(A,B) \\
Y &= A+B; \\
for i &= A:B \\
Y &= Y + i;
end \\
endfunction
\end{align*}
\]

In the program containing the function \( f2 \):

\[
\begin{align*}
function [ Y ] &= f2(X,Y) \\
X &= X .^ 2; \\
Y &= X .+ 2; \\
endfunction
\end{align*}
\]

Section 3: Write the output from the following segments of code exactly as it would appear it is executed in Matlab or Octave. Indicate blank spaces with an underscore. Assume that there are no errors. (4 points each)

(3.1) \[
\begin{align*}
a &= 2^4 / 4+4 * 2-2; \\
b &= 2^{(4/4+4)*2-2}; \\
c &= 27/9+64/4/4^2; \\
fprintf(' %3i
', a,b,c )
\end{align*}
\]
(3.2)  \[ a = \begin{bmatrix} 1:4 & 3:6 & 2,2,2 \end{bmatrix} ; \]
    \[ b = a(\text{end:end}); \]
    \[ c = a(\, a(1,2) \, , \, a(1,3) \, ); \]
    \[ d = a(\, : \, , \, 2 \, ); \]
    \[ \text{fprintf( '\%3i \%3i' , \, b, c, d \) } \]
% Note: there is 1 space between \%3i and \%3i

(3.3)  \[ x = [\, 100 : 2 : 10000 \, ]; \]
    \[ j = x(200) - x(198); \]
    \[ \text{for} \ i = 1:j \]
    \[ \quad \text{fprintf( '\%4i
' , x(i) ) \} \]
    \[ \text{end} \]

(3.4)  \[ i = 10; \, j = 3; \]
    \[ \text{while}(j < i) \]
    \[ \quad \text{fprintf( '%3i%3i' , i,j) \}
% Note: there are no spaces between %3i and %3i
    \[ \quad i = i - 7; \]
    \[ \quad j = j - 1; \]
    \[ \text{end} \]

(3.5)  \[ \text{hour} = 'ten'; \, \text{ten} = 'hour'; \, \text{one} = 'clock'; \]
    \[ x = [3 \, 5 \, 8 \, 2 \, 4]; \]
    \[ \text{switch} \ \text{hour} \]
    \[ \quad \text{case}'one', \text{ten} \]
    \[ \quad \quad \text{if}(x(2) > x(1)) \]
    \[ \quad \quad \quad \text{disp('koo koo')} \]
    \[ \quad \quad \text{elseif}(x(3) > x(2)) \]
    \[ \quad \quad \quad \text{disp('ding dong')} \]
    \[ \quad \quad \text{end} \]
    \[ \quad \text{case}'ten', \text{one} \]
    \[ \quad \quad \text{if}(x(4) == 2) \]
    \[ \quad \quad \quad \text{disp('bing bong')} \]
    \[ \quad \quad \text{elseif}(x(5) == 4) \]
    \[ \quad \quad \quad \text{disp('tik tok')} \]
    \[ \quad \quad \text{end} \]
    \[ \quad \text{otherwise} \]
    \[ \quad \quad \text{disp('wristwatch')} \]
    \[ \text{end} \]
(3.6) \[ a = [7.2, 4.3, 0.1, 9.9, 100.1, 70.0]; \]

\[
\text{for } i=1:2 \\
\quad j = 1; \\
\quad \text{while} (i < a(j)) \\
\quad \quad \text{fprintf}('%5.2f%6.1f\n', i,a(j) ) \\
\quad \quad j = j+1; \\
\quad \text{end} \\
\text{end} \\
% \text{Note: there are no spaces between } %5.2f \text{ and } %6.1f
\]

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:
\[
x = [\ 2:2:8\ ];
\]
\[
y = [\ 3:3:12\ ];
\]
\[
\text{fprintf}(' %2i\n', y(3:4) )
\]
\[
[ x, y ] = f1( x(1:2) , x(3:4) ); \\
\text{fprintf}(' %2i%2i\n', x, y )
\]
% \text{Note: there are no spaces between } %2i \text{ and } %2i

In the m-file containing the function f1:
\[
\text{function } [ a , b ] = f1( a , b )
\]
\[
a = b;
\]
\[
b = a;
\]
\endfunction
In the “main” program:

```plaintext
a = 2;
b = 4;
x = 10;
y = 12;
fprintf( '%3i%3i\n', a, b )
% Note: there are no spaces between %3i and %3i
[a, b] = f2( a, b, x, y );
fprintf( '%2i%2i\n', a, b )
% Note: there are no spaces between %2i and %2i
```

In the m-file containing the function f2:

```plaintext
function [ x, y ] = f2( y, x, a, b )
for i = 1:5
    x = x + 1;
y = y + 1;
    if(i==3)
        fprintf( '%2i%2i\n', a, b )
        return
    end
    a = a - 1;
b = b - 1;
end
defunction
```
(3.9)
In the “main” program:

\[ Y = @(X) X.^2 + 1; \]

PIG = 2;
COW = [3, 4, 5];
HORSE = DOG(PIG) - CAT(COW);
GOAT = DOG(COW) - CAT(PIG);
fprintf('%3i', HORSE, GOAT, Y(PIG.^2))

In the m-file containing the function DOG:

function [output] = DOG(X)
if(numel(X)>1)
    output = X(end);
    return
else
    output = X - 1;
    return
end
defunction

In the m-file containing the function CAT:

function [output] = CAT(Y)
output = 0;
for i=1:numel(Y)
    output = output + Y(i);
end
defunction
(3.10)
In the “main” program:
\[
x = 5;
y = (1:3);
w = f5(f6(x), y) + f6(y);
\]
\[
\text{fprintf('}'3i\n', w)
\]

In the m-file containing the function \texttt{f5}:
\[
\text{function } [y] = f5(w, t)
y = f6(w) + f6(t);
\text{fprintf('}'3i\n', y)
endfunction
\]

In the m-file containing the function \texttt{f6}:
\[
\text{function } [w] = f6(x)
\text{for } i=1:\text{numel}(x);
\quad x(1) = x(1) + x(i);
\text{end}
w = x(1);
\text{fprintf('}'3i\n', w)
endfunction
\]
Section 4: Write a program to complete the following tasks. Do not write the output of the code.

(4.1) (5 points) A program calls a function called `plotty`:

```python
plotty(x, y, w, t)
```

Write the function `plotty` that takes as input the arrays `t, x, y,` and `w` (there is no output). The function `plotty` displays the following information on three separate subplots.

1. In the upper half of the plotting window, plot `w` versus `t`.
2. In the lower left corner of the plotting window, plot `x` versus `t`.
3. In the lower right corner of the plotting window, plot `y` versus `t`.
(4.2) (5 points) A program calls a function called `trans`:

\[ AT = \text{trans}(A) \]

Write the function named `trans` that takes as input an array `A` and returns as output the transpose of `A` (called `AT`). For example,

<table>
<thead>
<tr>
<th>if A contains</th>
<th>AT would contain</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 4 7</td>
<td>1 8 0 1</td>
</tr>
<tr>
<td>8 9 0</td>
<td>4 9 0 4</td>
</tr>
<tr>
<td>0 0 0</td>
<td>7 0 0 6</td>
</tr>
<tr>
<td>1 4 6</td>
<td></td>
</tr>
</tbody>
</table>

Your function must be general for any sized `A` array. You cannot use any of Matlab and Octave’s built-in functions for this problem except functions that determine the number of elements, rows, or columns in an array. You cannot use the transpose operator in this problem.
(4.3) (5 points) A program calls a function named `coinflip`:

```python
results = coinflip(numflip)
```

Write the function named `coinflip` that takes as input the number of times (`numflip`) the user will flip a trick coin, and returns as output the result from each flip (`results`). The trick coin is weighted so that there is a 75% chance of heads and 25% of tails. If the coin lands on heads, the result for that flip is "H". If the coin lands on tails, the result for that flip is "T".

For example, if `numflip` is 5, three possible values for `results` are HHTHT, TTHHH, and HHHHH (there are many more possible outcomes). The actual values stored in `results` will depend on how the coinflips turn out.

Your function must be general for any value of `numflip`. 
(4.4) (6 points) A restaurant only sells two items, hamburgers and sodas. Each time a customer places an order, the number of hamburgers and sodas each customer orders is saved in a file called “customers.txt”. Yesterday, 157 customers visited the restaurant and the following data were recorded:

In “customers.txt”
1  1
2  0
... <rows 3-156 not shown> ...
1  4

The first column contains the number of hamburgers ordered by each customer. The second column contains the number of sodas ordered by each customer. For example, customer #157 ordered 1 hamburger and 4 sodas.

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

(1) How many hamburgers were ordered that day?
(2) Which customer (1-157) ordered the most sodas?
(3) How many customers ordered more sodas than hamburgers?

You cannot use any of Matlab and Octave’s built-in functions for this problem except functions that determine the number of elements, rows, or columns in an array.