Question 1: (20 points)

Part (a): (12 points)
Suppose the following fragment has been executed:

```matlab
% The first interval [a1,b1] has these endpoints:
a1 = rand(1); b1 = a1+rand(1);
% The second interval [a2,b2] has these endpoints:
a2 = rand(1); b2 = a2+rand(1);
% Assume a1, b1, a2, and b2 are unique.
```

(i) Complete the following fragment so that it prints ‘Yes’ if the second interval is inside the first interval and ‘No’ otherwise.

```
if __________________ a1<a2 && b2<b1
  disp('Yes')
else
  disp('No')
end
```

(ii) Complete the following fragment so that it prints ‘No’ if the the intervals fail to intersect and ‘Yes’ otherwise.

```
if ___________________ b2<a1 || b1<a2
  disp('No')
else
  disp('Yes')
end
```

Non-intersecting scenario 1: —[a1—b1]—[a2—b2]—
Non-intersecting scenario 2: —[a2—b2]—[a1—b1]—

Part (b): (8 points)
Write the loop condition below so that the fragment keeps prompting the user to enter a number until the value entered is positive and is a multiple of 3 or 5.

```matlab
n = input('Enter a number: ');
while ______________________________
  n = input('Enter a number: ');
end
```

Solution:

```matlab
n<=0 || rem(n,3)^=0 & rem(n,5)^=0
```

% Parentheses necessary since && has higher precedence than ||, but don’t take points off this time
Question 2: (10 points)

Part (a): (3 points)
What is the last line of output after executing the following fragment?

```plaintext
x = 2;
y = x*3;
while x<=6 && y<=6
   x = x + 2;
   disp(x)
end
```

*Answer:*

8

Part (b): (7 points)
The following fragment calculates and displays the first few Fibonacci numbers. When the fragment
finishes execution, which Fibonacci numbers are stored in variables `f_old`, `f_cur`, and `f_new`? You can,
but don’t have to, evaluate the Fibonacci numbers. For example, you can write `f_4` instead of its value 3.

```plaintext
n = 2;
f_old = 1  % f(1)
f_cur = 1  % f(2)
for n = 3:5
   f_new = f_old + f_cur
   f_old = f_cur;
   f_cur = f_new;
end
```

* f_old: 3, f_4  f_cur: 5, f_5  f_new: 5, f_5

*Note: Need 2 out of 3 correct to get partial credit*
Question 3: (20 points)

A certain bacteria has a growth rate that is dependent on the ambient temperature. At or below 32°F, there is no growth. Above 32°F the growth rate follows the formula

\[ aT^2 + b \]

where \( T \) is ambient temperature in °F, and \( a = 0.01 \) and \( b = -10 \) are model parameters. When the temperature is very high, above 90°F, the rate estimated by the above formula must be corrected by a reduction of 10%.

Complete the fragment below to compute and display the growth rate.

```matlab
T = input('What is the temperature? ');

% Calculate and display the growth rate of the bacteria

a = 0.01;  \% model parameter, ok if student doesn’t name this
b = -10; \% model parameter, ok if student doesn’t name this

if T <= 32
    rate = 0;
else
    rate = a*T^2 + b;
end

% Correct rate if necessary
if (T > 90)
    rate = rate*0.9;
end

fprintf('BAX has growth rate %f
', rate)
\% Any print format is ok
```

Do not write redundant (or useless) if or else branches; we took off points. See examples below.

```matlab
a= 0;
if x<y
    a= rand(1); \% OK
elseif x==z
    a= a; \% BAD
else
    \% BAD
end
```
Question 4: (20 points)

A unit hexagon centered at \((a, b)\) has vertices

\[
\begin{align*}
P_1 &: (a + \Delta x, b + \Delta y) \\
P_2 &: (a - \Delta x, b + \Delta y) \\
P_3 &: (a - 1, b) \\
P_4 &: (a - \Delta x, b - \Delta y) \\
P_5 &: (a + \Delta x, b - \Delta y) \\
P_6 &: (a + 1, b)
\end{align*}
\]

where \(\Delta x = 1/2\) and \(\Delta y = \sqrt{3}/2\). Assume that the function \texttt{DrawHex}(a,b) adds to the figure window a unit hexagon with center at \((a, b)\).

We say that a unit hexagon is “good” if it is entirely inside a square with vertices \((0,0)\), \((10,0)\), \((10,10)\), and \((0,10)\). Write a program fragment to randomly choose points from a square with vertices \((0,0)\), \((10,0)\), \((10,10)\), and \((0,10)\)—each coordinate is uniformly random in the interval \((0,10)\). Whenever there is a point that can be the center of a \textit{good} hexagon, draw the hexagon. Your fragment should draw exactly 100 good hexagons. Do not write code to set up the figure window and axes.

\[
deltaY = \sqrt{3}/2; \quad \% \text{OK if student doesn’t name a constant}
\]

\[
k = 0;
\]

\[
\text{while} \quad k < 100
\]

\[
\%
\text{Draw the } k\text{-th good hexagon}
\]

\[
a = 10*\text{rand}(1);
\]

\[
b = 10*\text{rand}(1);
\]

\[
\text{if} \quad 0<a-1 \&\& a+1<10 \&\& 0<b-\delta Y \&\& b+\delta Y<10
\]

\[
\%
< \text{instead of } \leq \text{ is OK; check } a, b \text{ separately OK}
\]

\[
\text{DrawHex}(a,b)
\]

\[
k = k+1;
\]

\[
\text{end}
\]

\[
\%
\text{An alternate solution} \quad \%
\]

\[
\text{for } k = 1:100
\]

\[
a = 10*\text{rand}(1);
\]

\[
b = 10*\text{rand}(1);
\]

\[
\text{while } a<1 \; \| \; a>9 \; \| \; b<\delta Y \; \| \; b>10-\delta Y
\]

\[
a = 10*\text{rand}(1);
\]

\[
b = 10*\text{rand}(1);
\]

\[
\%
\text{Check } a \text{ and } b \text{ separately OK}
\]

\[
\text{DrawHex}(a,b)
\]

\[
\text{end}
\]
Question 5: (30 points)

A unit hexagon has six unit hexagon neighbors with these centers

\[ H_1 : (a + 3 \Delta x, b + \Delta y) \]
\[ H_2 : (a, b + 2 \Delta y) \]
\[ H_3 : (a - 3 \Delta x, b + \Delta y) \]
\[ H_4 : (a - 3 \Delta x, b - \Delta y) \]
\[ H_5 : (a, b - 2 \Delta y) \]
\[ H_6 : (a + 3 \Delta x, b - \Delta y) \]

where \( \Delta x = 1/2 \) and \( \Delta y = \sqrt{3}/2 \). Assume that the function \texttt{DrawHex(a,b)} adds to the figure window a unit hexagon with center at \((a,b)\).

Complete the fragment below to draw \( K \) columns of a “slanted” bee hive. Each column is made up of \( n \) unit hexagons. Center the top left hexagon on the origin \((0,0)\). An example with 5 hexagons in each of 3 columns is shown below. Do not write code to set up the figure window and axes.

```matlab
n = input('How many hexagons in each column? ');  
K = input('How many columns? ');  

% Draw a slanted bee hive with n hexagons in each  
% of K columns

% OK if student doesn't name these constants
xdist = 3/2;  % x-dist btw hex ctrs in adjacent cols  
ydist = sqrt(3);  % y-dist btw hex ctrs in a column  
deltaY = sqrt(3)/2;

for c = 1:K  
  % In column c...  
  x = (c-1)*xdist;  
  yOffset = -(c-1)*deltaY;  
  for r = 1:n  
    % The rth hexagon...  
    y = yOffset - (r-1)*ydist;  
    DrawHex(x,y)  
  end  
end  

% An alternate solution %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nShift = 0;  % How many deltaY's to shift down
for x = 0 : xdist : (K-1)*xdist  
  ystart = nShift*deltaY;  
  for y = ystart : -ydist : ystart-(n-1)*ydist  
    DrawHex(x,y)  
  end  
  nShift = nShift-1;  
end
```