Question 1: (15 points)

(a) What is the output from executing the following script? If the program doesn’t terminate or if there will be an error during execution, write the word “error” instead of the output.

```matlab
v = [3 1 4 2];
for k = 1:4
    v(v(k)) = v(k);
end
disp(v)
```

Solution:

```
1 2 3 2
```

(b) What will be printed when the following script is executed? Use the specified print format.

<table>
<thead>
<tr>
<th>Script</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x = 9;</code></td>
<td></td>
</tr>
<tr>
<td><code>y = 4;</code></td>
<td></td>
</tr>
<tr>
<td><code>z = ha(y,x);</code></td>
<td></td>
</tr>
<tr>
<td><code>fprintf('z is %d\n', z)</code></td>
<td></td>
</tr>
<tr>
<td><code>v = x - y;</code></td>
<td></td>
</tr>
<tr>
<td><code>fprintf('x is %d\n', x)</code></td>
<td></td>
</tr>
<tr>
<td><code>fprintf('x is %d\n', x)</code></td>
<td></td>
</tr>
<tr>
<td><code>fprintf('y is %d\n', y)</code></td>
<td></td>
</tr>
<tr>
<td><code>fprintf('y is %d\n', y)</code></td>
<td></td>
</tr>
</tbody>
</table>

Solution:

```
x is 12
y is 9
z is 3
x is 9
y is 4
```

Part (c): Assume function `randomWalkDist` has been implemented correctly and is accessible.

```matlab
function d = randomWalkDist(n,x0,y0)
% d is the distance from (x0,y0) after n steps of a random walk.
```

Below are five attempts to call function `randomWalkDist` to determine the distance from the point (1,2) after 100 steps of a random walk. Which function calls are correct? Write in each box ‘C’ for correct or ‘N’ for incorrect.

1. C disp(randomWalkDist(100, 1, 2))
2. N d = randomWalkDist(100, [1, 2])
3. N distance = randomWalkDist(100, (1, 2))
4. N distance = randomWalkDist(100, 1:2)
5. C distance = randomWalkDist(100, 1, 2)
Question 2: (20 points)

(a) Write a boolean expression on the blank below so that the resulting fragment keeps prompting the user to enter a number until an appropriate number is entered or until 10 numbers have been entered, whichever occurs first.

```matlab
n = input('Enter a value that is either negative or greater than 500: ');  
k = 1;  
while ~ ( n < 0 || n > 500 || k >= 10 )  
    n = input('Enter a value that is either negative or greater than 500: ');  
    k = k+1;  
end
```

EXAMPLE SOLUTIONS:

- \( n >= 0 \land n <= 500 \land k < 10 \)
- \( n >= 0 \land n <= 500 \land k != 10 \)
- \( \neg ( n < 0 \lor n > 500 \lor k == 10 ) \)
- \( \neg ( n < 0 \lor n > 500 \lor k >= 10 ) \)

(b) Complete the fragment below to draw as many non-overlapping disks of diameter \( W \) as possible within an \( L \)-by-\( W \) rectangle, \( 0 < W < L/2 \). The disks are evenly distributed within the rectangle with the leftmost and rightmost disks tangent to the left and right sides of the rectangle, respectively. The leftmost disk is centered at (0,0).

Assume the availability of the function `DrawDisk`. For example, `DrawDisk(3,5,2,'w')` draws a white disk of radius 2 centered at (3,5). Do not use built-in function `linspace`.

```matlab
figure; axis equal off; hold on;  
L = input('What is the length L of the rectangle? ');  
W = input('What is the width W of the rectangle (W<L/2)? ');  
r = W/2; % radius of each disk  
% Write your code below to draw the disks as specified. DO NOT USE linspace.

n = ____ floor( L/W ) ____; % number of disks

g = ____ (L - n*W) / (n-1) ____; % gap between two disks

% c2c= (L-W)/(n-1); % center-to-center distance

for ____ k = 1 : n ____
    x = (k-1)*(g+W); % x= (k-1)*c2c
    DrawDisk( x, 0, r, 'w' )
end

hold off
```
Question 3: (20 points)

A certain script, when executed, produces the figure shown on the right. Write the code that should be added to such a script in order to plot an asterisk at a randomly generated location within the axes area and display one of the following messages in the Command Window:

- “Blue” if the asterisk is in the blue semicircle with radius 2
- “White” if the asterisk is in the white half-ring with outer radius 4
- “Yellow” if the asterisk is in the upper-right section colored yellow
- “Black” if the asterisk is in the lower-right section colored black
- “Gray” if the asterisk is in the gray rectangular area on the left

The random location of the asterisk should be generated such that the x-coordinate is a real number equally likely to be any value in the open interval (-4,5) while the y-coordinate is an integer equally likely to be any integer in the range 10 to 18, inclusive.

Reminders: The distance between two points \((x_1, y_1)\) and \((x_2, y_2)\) is \(\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}\). The command to plot a red asterisk at \((3,2)\) is `plot(3,2,'r*')`.

The only built-in functions allowed are `sqrt`, `rand`, `round`, `floor`, `ceil`, `plot`, `disp`, and `fprintf`.

% Solution
x= rand*9-4;
y= floor(rand*9 + 10); % y= ceil(rand*9 + 9);

plot(x,y,'r*') % Any color asterisk is ok

if x < 0
disp('Gray')
else
  d= sqrt(x^2 + (y-14)^2);
  if d < 2
    disp('Blue')
  elseif d < 4
    disp('White')
  elseif y > 14
    disp('Yellow')
  else
    disp('Black')
  end
end
Question 4: (20 points)

Given two vectors of the same length, we say that the two vectors are “friendly” if each pair of corresponding components of the vectors sum to the same value. For example, of the three vectors given below, b and c are friendly but c and d are not friendly.

\[
\begin{align*}
  b &= [ 6 13 1 0 19 ] \\
  c &= [ 10 3 15 16 -3 ] \\
  d &= [ 6 30 1 0 19 ]
\end{align*}
\]

Write a fragment below to determine whether the given vectors \( f \) and \( g \) are friendly. The fragment should print the word “friendly” if the two vectors are friendly; otherwise print the message “not friendly”. For full credit, use a \texttt{while}-loop to efficiently solve this problem. \textit{Do not use} any built-in function other than \texttt{length}. Do not use vectorized code.

\[
\begin{align*}
  \text{% Assume vectors } f \text{ and } g \text{ have been created. Vectors } f \text{ and } g \text{ have the same} \\
  \text{% length and the length is greater than 2. Write your code below.} \\
  \text{% DO NOT use any function other than length. DO NOT use vectorized code.}
\end{align*}
\]

Example Solution:

\[
\begin{align*}
  s &= f(1) + g(1); \\
  k &= 2; \\
  n &= \text{length}(f); \\
  \text{while } k <= n & \text{ and } f(k) + g(k) == s \\
    k &= k + 1; \\
  \text{end} \\
  \text{if } k > n & \text{ then } k = n + 1 \\
    \text{disp('friendly')} \\
  \text{else} \\
    \text{disp('not friendly')} \\
  \text{end}
\end{align*}
\]

%%%% Alternative while-loop and post processing

\[
\begin{align*}
  \text{% while } k <= n \\
  \text{% if } f(k) + g(k) == s \\
  \text{% k = k + 1;} \\
  \text{% else} \\
  \text{% k = n + 2;} \\
  \text{% end} \\
  \text{% if k == n+1} \\
  \text{% disp('friendly')} \\
  \text{% else} \\
  \text{% disp('not friendly')} \\
  \text{% end}
\end{align*}
\]
Question 5: (25 points)

(a) Implement the following function as specified. Recall that the distance between two points $(x_1, y_1)$ and $(x_2, y_2)$ is $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

```matlab
function [tf, dista] = disjoint(x1,y1,r1,x2,y2,r2)
% Determine the distance between the centers of two disks and whether the disks are disjoint. Two disks are disjoint if they do not overlap.
% x1, y1: the x-coordinate and y-coordinate of the center of disk 1
% x2, y2: the x-coordinate and y-coordinate of the center of disk 2
% r1, r2: the radii of disk 1 and disk 2, respectively
% tf: tf is true (1) if disks 1 and 2 are disjoint; otherwise tf is false (0)
% dista: the "center-to-center distance," i.e., distance between the centers of disks 1 and 2

% Solution:
dista= sqrt((x2-x1)^2 + (y2-y1)^2);
tf= dista > r1+r2; % >= is ok
```

(b) Given a set of blue disks and a set of green disks, consider all possible pairs that consist of one blue and one green disks. Implement the following function to find the maximum center-to-center distance among all those possible pairs. For full credit, make effective use of function disjoint (assume it has been implemented correctly and is accessible). Do not use built-in functions max or min.

```matlab
function maxDist(bx,by,gx,gy)
% Determine and print the maximum center-to-center distance from all possible pairs of disks that consist of one blue and one green disks.
% bx and by are vectors of the same length;
% bx(k) and by(k) are the x- and y-coordinates of the center of the kth blue disk.
% gx and gy are vectors of the same length;
% gx(i) and gy(i) are the x- and y-coordinates of the center of the ith green disk.
% The number of blue disks and green disks may be different.
% All disks have unit radius (radius is 1).
% DO NOT USE built-in functions max or min.

% Solution:
maxD= -1; % 0 also ok

for k= 1:length(bx)
    for i= 1:length(gx)
        [tf, d] = disjoint(bx(k),by(k),1,gx(i),gy(i),1);
        % [~, d] = disjoint(bx(k),by(k),1,gx(i),gy(i),1); also correct
        if d > maxD
            maxD= d;
        end
    end
end
disp(maxD)
```