Question 1: (10 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 = [5 3 1 2 6];
for k = 1:4
    v(k) = v(k+1);
    disp(v(k))
    if v(k)<k
        k = 9;
    end
end
```

Solution:

3
1
2
6

(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>x = 4;</td>
<td>function y = moo(x,z)</td>
</tr>
<tr>
<td>y = 8;</td>
<td>y = x+1;</td>
</tr>
<tr>
<td>z = moo(x+1,y);</td>
<td>x = z;</td>
</tr>
<tr>
<td>fprintf('z is %d\n', z)</td>
<td>z = y+1;</td>
</tr>
<tr>
<td>fprintf('x is %d\n’, x)</td>
<td>fprintf('x is %d\n’, x)</td>
</tr>
<tr>
<td>fprintf('y is %d\n’, y)</td>
<td>fprintf('z is %d\n’, z)</td>
</tr>
</tbody>
</table>

Solution:

x is 8
z is 7
z is 6
x is 4
y is 8
Question 2: (20 points)

(a) Fill in the blank below to assign to \( w \) a randomly generated value such that \( w \) is equally likely to be any real value in the open interval (-13,2).

\[
w = \ldots \text{rand*15 - 13} \ldots;
\]

(b) Given a vector \( v \) that has a length greater than 2, we define the “neighbor sums” to be the sums of adjacent (i.e., neighboring) values in \( v \). For example, the vector

\[
\begin{bmatrix}
6 & -3 & -5 & 1 & 4 & 1 & 2 & 0.1
\end{bmatrix}
\]

has the neighbor sums

\[
\begin{bmatrix}
3 & -8 & -4 & 5 & 5 & 3 & 2.1
\end{bmatrix}
\]

Write a fragment below to determine and print the largest neighbor sum of a given vector \( v \). In the example above, the largest neighbor sum is 5. DO NOT use any built-in functions other than \texttt{length}.

% Assume \( v \) is a vector of numbers and its length is > 2.
% Display the largest neighbor sum of vector \( v \).
% Write your code below. DO NOT use any built-in functions other than length.

Solution:

\[
\text{bestSoFar} = v(k)+v(k+1); \quad \% \text{ or init to -inf, but 0 is wrong}
\]

\[
\text{for } k = 2: \text{length}(v)-1 \quad \% \text{If bestSoFar was init to -inf}
\]

\[
\text{\quad \text{then k starts at 1}
\]

\[
\text{s} = v(k) + v(k+1);
\]

\[
\text{if } s > \text{bestSoFar}
\]

\[
\text{\quad bestSoFar} = s;
\]

end
end

\[
\text{fprintf(''}\text{Largest neighbor sum in } v \text{ is } %f\text{\n', }\text{bestSoFar}\text{')}
\]
Question 3: (20 points)

(a) Implement the following function as specified:

```matlab
function [vol, linLen] = boxFeatures(X, Y, Z)
% Return the volume and "linear length" of a rectangular box with length X,
% width Y, and height Z. X, Y, and Z are in meters.
% vol: the volume in cubic meters.
% linLen: the "linear length" in meters, which is the sum of the length,
% width, and height of the box.
vol = X*Y*Z;
linLen = X+Y+Z;
```

Solution:

```matlab
vol = X*Y*Z;
linLen = X+Y+Z;
```

(b) A courier determines the charge for shipping a rectangular box of goods as follows: A box heavier than 30kg or that has a volume greater than 2m$^3$ is not allowed. The base charge for an allowed box is $14. Every kilogram over 10kg is additionally charged $1.30. Furthermore, a box heavier than 10kg and that has a linear length greater than 2.5m incurs a $5 surcharge.

Complete the fragment below to display the charge for shipping a box. If the box is not allowed, display the words “not allowed”. Make effective use of the function `boxFeatures` from Part (a); assume that it is correctly implemented and accessible.

```matlab
L = input('Enter the length of the box in meters: ');
W = input('Enter the width of the box in meters: ');
H = input('Enter the height of the box in meters: ');
M = input('Enter the weight of the box in WHOLE kilograms (no fractional part): ');
% Write your code below.
[v, linLen] = boxFeatures(L, W, H);
if M > 30 || v > 2
    disp('not allowed')
else
    cost = 14;
    if M > 10
        cost = cost + (M-10)*1.3;
        if linLen > 2.5
            cost = cost + 5;
        end
    end
    disp(cost)
end
```

Alternative if-statement:

```matlab
if M > 30 || v > 2
    disp('not allowed')
elseif M > 10 && linLen > 2.5
    disp(14 + (M-10)*1.3 + 5)
elseif M > 10
    disp(14 + (M-10)*1.3)
else
    disp(14)
end
% disp done within the branches, not here
```

Grading note when variable created inside some but not all branches of an if-statement:

```matlab
if x>30
    disp(x)
else
    cost = x;
end
disp(cost)  % This causes an error if x is >30 because then variable cost doesn’t exist
```
Question 4: (25 points)

Implement the function below as specified.

function [x, y] = myRandomWalk(xf, yf, u)
% Perform a random walk starting from position (0,0). In each step, move north or
% east one unit and it is u times as likely to move north as to move east. The
% walk ends when position (xf,yf) is reached or when 90 steps have been taken,
% whichever happens first.
% Vectors x and y store the path of the walk such that (x(k),y(k)) is the position
% BEFORE the kth step of the walk, i.e., x(1) and y(1) should store the starting
% point. Assume position (xf,yf) is different from the starting point.
% DO NOT use 2-d array.

Solution:

x(1)= 0;
y(1)= 0;
k= 1; % Next step number

while ( x(k)~=xf || y(k)~=yf ) && k<=90
    r= rand;
    if  r < 1/(u+1) % go east
        dx= 1;
dy= 0;
    else % go north
        dx= 0;
dy= 1;
    end
    x(k+1) = x(k) + dx;
y(k+1) = y(k) + dy;
    k = k + 1;
end
Question 5: (25 points)

Complete the fragment below to draw an upside down brick pyramid. Each brick has height 1 and its width is user-entered. The number of bricks in the top row is user-entered and the top leftmost brick has its lower left corner at (0,0). Each row has one fewer brick than the row above and the bottom row has one brick only. Each row is centered relative to the top row.

Assume the availability of the function `DrawRect` and use it to draw each brick. For example, the command `DrawRect(5,7,3,1,'y')` draws a yellow rectangle that has width 3 and height 1 with its lower left corner at (5,7). An example figure is shown on the right.

\[\text{n = input('How many bricks will be in the top row of the pyramid? ')};\]
\[\text{w = input('How wide is one brick? ')};\]
\[\text{figure; axis equal; hold on}\]
\[\text{\% Write your code below.}\]
\[\text{Example solution 1 :}\]
\[\text{y= 0;}\]
\[\text{for r= n:-1:1} \quad \% \text{Row n is top row, i.e., row 1 is bottom row.}\]
\[\text{\% The rth row of blocks (row r has r blocks)}\]
\[\text{x= (n-r)*w/2; \quad \% \text{left end of block}\}
\[\text{for k= 1:r}\]
\[\text{\quad DrawRect(x,y,w,1,'y')}\]
\[\text{\quad x= x+w;}\]
\[\text{\quad end}\]
\[\text{\quad y= y-1;}\]
\[\text{end}\]
\[\text{title(sprintf('Top row has %d bricks. Each brick has width %.1f.', n,w))}\]
\[\text{hold off}\]
\[\text{Example solution 2:}\]
\[\text{for r= 1:n} \quad \% \text{Row 1 is top row}\]
\[\text{y= 1-r;}\]
\[\text{x= (r-1)*w/2; \quad \% \text{At row r x starts at r-1 half-bricks from axis}\}
\[\text{for k= 1:n-r+1} \quad \% \text{There are n-r+1 bricks in row r}\]
\[\text{\quad DrawRect(x,y,w,1,'y')}\]
\[\text{\quad x= x + w;}\]
\[\text{\quad end}\]
\[\text{end}\]