ENGR 132 Exam Practice
Nested Structures

Problems

Practice 1.

Write a MATLAB script that creates a 12 x 8 matrix A with the following criteria.

a. The elements of the first row equal twice the value of the column in which they are located.

b. The elements of the first column equal twice the value of the row in which they are located.

c. The rest of the elements are equal to the difference between the element immediately above them and the element immediately to the left of them. (Ex: 4-4 = 0 and 6-0=6)

A partial matrix A is shown below.

$$A = \begin{bmatrix} 2 & 4 & 6 & \cdots \\ 4 & 0 & 6 & \\ 6 & -6 & 12 & \\ \vdots & & & \ddots \end{bmatrix}$$

You must use *nested for loops* and *a selection structure* to programmatically generate the matrix. Build the matrix by populating the first column, then fill in the rest of the first row, then the rest of the second row, and so on. The first line of code, creating an empty matrix A, has been provided for you.

Solution

Practice 2.

A nested loop has been coded as shown below:

```
for a = 2:3:6
   for b = 1:2
       c = a-b;
       fprintf('%.0f %.0f %.0f\n',a,b,c)
   end
end
```

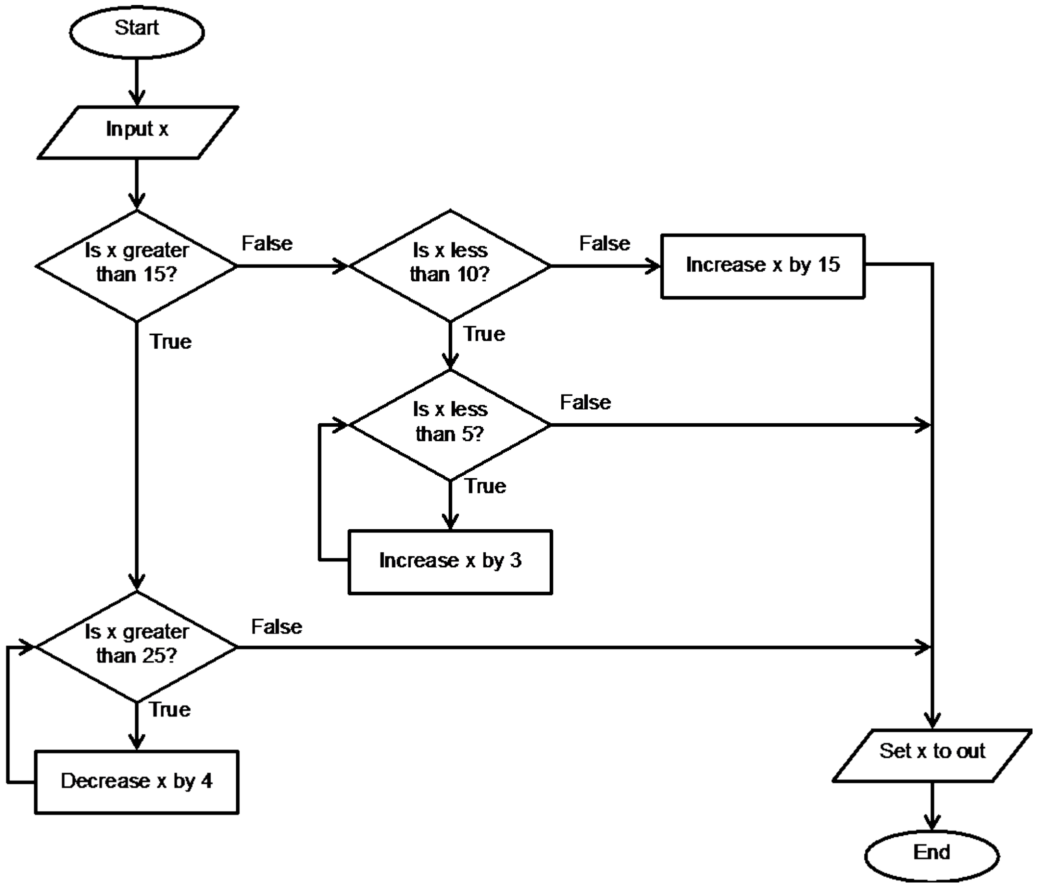
Fill in all of the columns in the variable tracking table below for the next 3 iterations of the looping structure.

Iteration (outer-inner)	Iteration	a	b	c
0-0	0	--	--	--
1-1	1	2	1	1
	2			
	3			
	4			

Solution

Practice 3.

Consider the following flowchart:



To help you test your flowchart, your friend Kono gives you two test cases: **`x = 12`** and **`x = 20`**. In addition to these two test cases, come up with two additional test cases (use integers) that test different aspects of the flowchart ***that have not been considered by the two test cases already*** provided. For each test case, state the purpose of the test case and give the final value of the variable **`out`**.

	Test Case	Purpose of Test Case	Value of out
1	x = 12		out =
2	x = 20		out =
3	x =		out =
4	x =		out =

Solution

Practice 4.

Track the variables n, k, and a through the execution of the MATLAB code shown below in the variable tracking table provided on the answer sheet. Every time vector a is changed, write out the value for n, the value for k, and the full a vector (as shown on the first line of the table).

```
n = 1;
k = 0;
a = [2 7];

while n <= length(a)
      if a(n) < 5
            for k = 1:2
                  a(n) = a(k) + k;
            end
      else
            a(n) = n;
      end
      n = n + 1;
end
```

Tracking table on Answer Sheet

n	**k**	**a**
1	0	2 7

Solution

Practice 5.

The following code has been written in MATLAB. What will be displayed to the screen?

```
x = -1;
y = -3;
for counter1 = 1:2:4
    for counter2 = 4:-2:1
        x = x * counter2;
        y = y + counter1;
        fprintf('%3i %3i %3i %3i \n',counter1,counter2,x,y)
    end
end
```

Solution

Practice 6.

After running the command `A = zeros(5,5)`, you have a 5x5 matrix with all zeros. Now use two nested **for loops** to set the value of each element in the matrix to equal the sum of the row index and column index. Your final A matrix should be:

$$A = \begin{bmatrix} 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \\ 5 & 6 & 7 & 8 & 9 \\ 6 & 7 & 8 & 9 & 10 \end{bmatrix}$$

Solution

Practice 7.

The following function transposes a matrix by converting its rows to columns as in the example below.

```
A =

     1     2     3     4
     5     6     7     8
     9    10    11    12

MatrixTransposed =

                          1     5     9
                          2     6    10
                          3     7    11
                          4     8    12
```

Complete lines A, B, and C of the MATLAB user-defined function called **`transposeAMatrix.`** This user-defined function has one input, called **`MatrixInput`**. The output of the user-defined function is a matrix that has been transposed and stored in an output variable called **`MatrixTransposed`**.

```
function MatrixTransposed = transposeAMatrix (MatrixInput)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% transposeAMatrix - This function transposes a matrix
% INPUT
%     MatrixInput – a matrix to be transposed
% OUTPUT
%     MatrixTransposed - the transpose of MatrixInput
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Assign the number of rows to variable nRows
% and the number of columns to variable nCols
[nRows, nCols] = size(MatrixInput);

% Loop through rows
for loopRows = _________________________ % LINE A
    % loop through columns
    for loopCols = _________________________ % LINE B

       % Compute the transpose
       MatrixTransposed(______,______)=MatrixInput(______,______); % LINE C

    end
end
```

Solution

Practice 8.

Your paired programming partner started the program below for you to finish.

```
function [] = find_negatives(score_array)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function to count the number of negative scores in each column of an
% array. The function will then print to the screen "Column X has Y
% negative values" where X = column number and Y equals number of negative
% scores in the column.
%
% Inputs
% 1. An array of scores
%
% Outputs
% None
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dimensions = size(score_array);
rows = dimensions(1);
columns = dimensions(2);
```

```
Part A.   for ____________________________________________  % outer loop
              negatives_count = 0; %set count of negative values to 0
Part B.       for ____________________________________________  % inner loop
Part C.           %1. Count the number of negative values in each column.
                  %2. Use fprintf to display "Column X has Y negative values" where X =
                  % column number and Y equals number of negative scores in the column
```

Given the function description and comments, you need to finish the nested loops by completing the loop control statements for both loops (Parts A and B) and completing remainder of the program (Part C).

Solution

Practice 9.

Given the code:

```
total = 0;
for a = 1:2:4
    delta = a;
    for b = 1:1:3
        total = total + 2*delta;
    end
end
```

Complete the tracking table on the answer sheet, stopping when the program would end. Each row in the table records values at the end of that iteration. (Note: you may not need all the rows provided)

Answer Sheet

Iteration	a	delta	b	total
0				

Solution

Solutions

Practice Solution 1

```
A = [ ];
for nrow = 1:12   (OR 1:8)
     for ncol = 1:8     (OR 1:12)
          if ncol ==1
            A(nrow,ncol) = 2*nrow
         elseif nrow ==1
            A(nrow,ncol) = 2*ncol
         else
            A(nrow,ncol) = A(nrow - 1, ncol) - A(nrow, ncol -1)
         end
       end
end

% Alternative
A(1,:) = 2*[1:12];   (OR 2*[1:8] if swapped numbers)
for nrow = 1:12   (OR 1:8)
     for ncol = 2:8     (OR 2:12)
         if nrow ==1
            A(nrow,ncol) = 2*ncol
         else
            A(nrow,ncol) = A(nrow - 1, ncol) - A(nrow, ncol -1)
         end
       end
end

% Alternative
for row = 1:12    (OR 1:8 if swapped numbers)
    A(1,row) = 2*row;
end
(remainder of ALT2 uses same nested structure as ALT1)
```

Practice Solution 2

Iteration (outer-inner)	Iteration	a	b	c
0-0	0	--	--	--
1-1	1	2	1	1
1-2	2	2	2	0

2-1	3	5	1	4
2-2	4	5	2	3

Practice Solution 3

	Test Case	Purpose of Test Case	Value of out
1	`x = 12`	This test case returns false to the first and second decisions of the flowchart, so its value is increased by 12.	out = 27
2	`x = 20`	This test case returns true to the first decision, but false to the next decision and does not enter the while loop and never changes its value.	out = 20
3/4	`x > 25`	Test cases of these values will return true to the first decision and the next decision will enter the while loop.	out = 25, 24, 23, or 22 (Based on initial x)
3/4	`5 <= x < 10`	Test cases of these values will return a false to the first decision, a true to the second decision, but false to the third decision and will not enter the while loop. They will not change their value.	out = 5, 6, 7, 8, or 9 (Same as initial x)
3/4	`x < 5`	Test cases of these values will return a false to the first decision, a true to the second and third decisions and enter the while loop.	out = 5, 6, or 7 (Based on initial x)

Practice Solution 4

n	k	a
1	0	2 7
1	1	3 7
1	2	9 7
2	2	9 2
3	--	--
3	2	9 2

Practice Solution 5

```
1     4     -4    -2
1     2     -8    -1
3     4     -32    2
3     2     -64    5
```

Practice Solution 6

```
for r = 1:5
    for c = 1:5
        A(r,c) = r+c;
    end
end
```

Practice Solution 7

Line A.	`for loopRows = 1:nRows`
Line B.	`for loopCols = 1:nCols`
Line C.	`MatrixTransposed( loopCols , loopRows ) = MatrixInput( loopRows , loopCols )`

Practice Solution 8

```
function [] = find_negatives(score_array)
% See exam booklet for function description

dimensions = size(score_array);
rows = dimensions(1);
columns = dimensions(2);
```

Part A

```
for c = 1:1:columns % outer loop
    negatives_count = 0; %set count of negative values to 0
```

Part B

```
    for r = 1:1:rows % inner loop
```

Part C

```
        % 1. Count the number of negative values in each column.
        % 2. Use fprintf to display "Column X has Y negative values" where X =
        % column number and Y = number of negative scores in the column.
        if score_array(r, c) < 0;
            negatives_count = negatives_count + 1;
        end
    end
    fprintf('Column %.0f has %.0f negative values \n', c, negatives_count)
end
```

Practice Solution 9

Iteration	a	delta	b	total
0	-	-	-	0
1 (or 1 – 1)	1	1	1	2
2 (or 1 – 2)	1	1	2	4
3 (or 1 – 3)	1	1	3	6
4 (or 2 – 1)	3	3	1	12
5 (or 2 – 2)	3	3	2	18
6 (or 2 – 3)	3	3	3	24