ENGR 132 Exam Practice
Indexed Structures

Problems

Practice 1.

The Fibonacci Sequence is a sequence of numbers that looks like this:

1, 1, 2, 3, 5, 8, 13…

This pattern is generated such that the nth term is the sum of the two previous terms: the (n-1)th term and the (n-2)th term.

Fill in the part of the script below that builds the vector, `Fib_seq`, with the elements of the Fibonacci Sequence. The sequence should end with the first element that exceeds the set threshold value (i.e., last_term_threshold).

```
% Hard code in positive integer value for the last value threshold
last_term_threshold = 1000;

% Initialize Fib_seq with first two elements
Fib_seq = [1 1]

% Initialize the counter
counter = 3;

% Write  the code necessary to produce the Fibonacci Sequence such the
% sequence ends once the last term is more than last_term_threshold

% ADD YOUR CODE HERE

% Display the resultant vector to the command window.
disp(Fib_seq)
```

Solution

Practice 2.

`Vect_A` is a vector of integers. The following code sorts the integers in numerical order into a new vector called `A_sorted`. Track the variables through the iterations of the for loop. The tracking table may include more iterations than needed.

NOTE: Write out the full vectors for `position_A` and `A_sorted`.

```
% Hard code Vect_A
Vect_A =  [1 2 0 -1]

% For loop to sort Vect_A into numerical order
% (smallest to largest)

for index = 1:length(Vect_A)
    position_A(index) = find(Vect_A==(min(Vect_A) + index-1));
    A_sorted(index)=Vect_A(position_A(index));
end
```

Answer Sheet:

Iteration	**index**	`position_A` **(show the full vector)**	`A_sorted` **(show the full vector)**
0			
1			
2			
3			
4			
5			
6			
7			

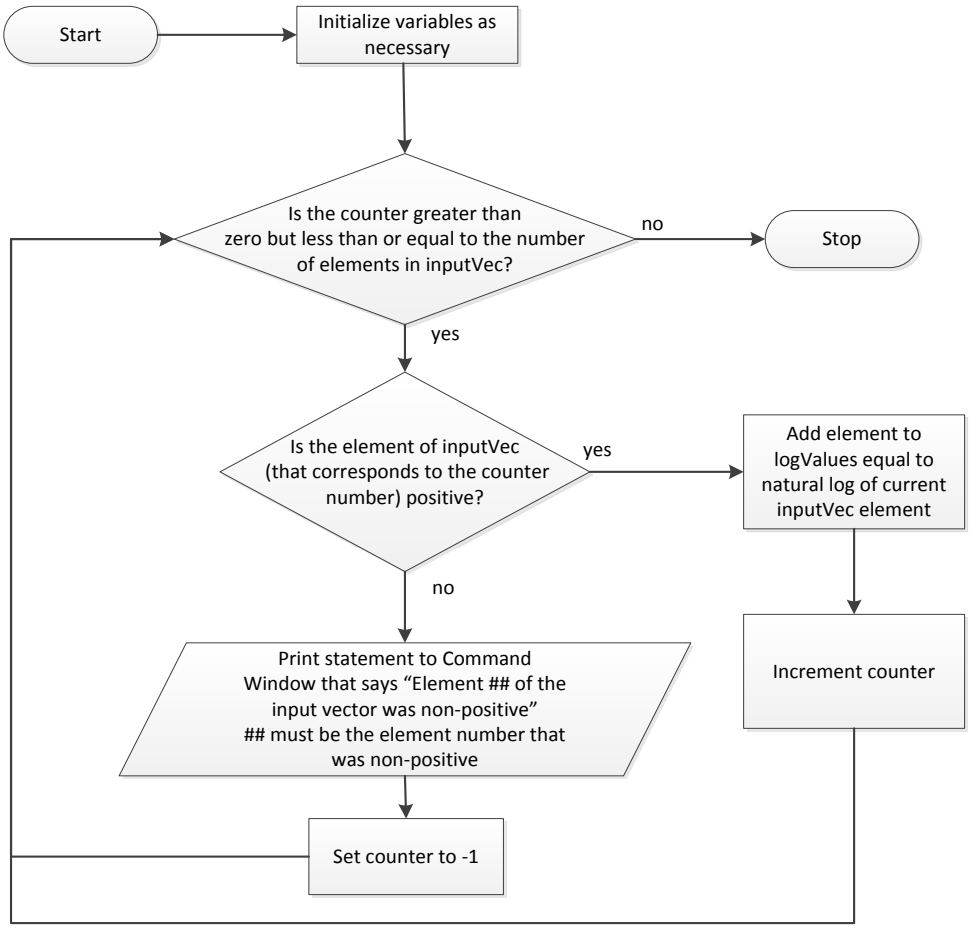
Solution

Practice 3.

A user-defined function called `logTrimmer` takes as input a vector of numeric values named `inputVec`. The function returns a vector of values using the vector variable named `logValues`. The function will read through inputVec. It will find the natural log of each positive, non-zero number in the vector one at a time, in the order that they appear in the vector.

If a negative or zero number is encountered, then the function will stop calculating the log values and will display a printed statement to the Command Window indicating the location of the first invalid number in `inputVec`. Do not use the error command to display the text; the function should return whatever log values were calculated prior to encountering a non-positive input value.

A. Write the user-defined function `logTrimmer` according to the given information and the flowchart below. Do not include header comments, but do comment your lines of code appropriately.



A. Run the code in the flowchart by calling `y = logTrimmer(x)` in MATLAB's Command Window, where `x = [3 4 5 0 -1]`. The `x` vector has invalid values. What print statement displays to the Command Window?

B. If, after you run Step B, you type `length(y)` in the command window, what will MATLAB output?

Solution

Solutions

Practice Solution 1

```
while Fib_seq(counter-1)<= last_term_threshold
    Fib_seq(counter) = Fib_seq(counter-1) + Fib_seq(counter-2)
    counter = counter + 1
end
```

OR alternative way of growing the sequence vector

```
    new_term=Fib_seq(counter-1)+Fib_seq(counter-2)
    Fib_seq=[Fib_seq new_term]
```

OR alternative logic statements

```
while Fib_seq(length(Fib_seq))<= last_term_threshold

while Fib_seq(end)<= last_term_threshold

while(max(Fib_seq)<=last_term_threshold
```

Practice Solution 2

Iteration	index	`position_A` (show the full vector)	`A_sorted` (show the full vector)
0	--	--	--
1	1	[4]	[-1]
2	2	[4 3]	[-1 0]
3	3	[4 3 1]	[-1 0 1]
4	4	[4 3 1 2]	[-1 0 1 2]
5			
6			
7			

Practice Solution 3

Part A

```
function [logValues] = logTrimmer(inputVec)

counter = 1; % initialize counter
num_values = length(inputVec); % find length of input vector

% while loop to run through the elements of the input vector
while counter > 0 & counter <= num_values
      % if-else structure to check whether element is valid. If valid,
      % it takes the natural log of the element; if not valid, it
      % prints that the element is not valid to the command window.
      if inputVec(counter) > 0
            % take natural log of element
            logValues(counter) = log(inputVec(counter)); % take ln of element
            counter = counter + 1; % advance counter
      else
            % print the error statement
            fprintf('Element %.0f of the input vector was non-positive.\n',
            counter)
            counter = -1; % make counter invalid
      end
end
```

B.

```
Element 4 of the input vector was non-positive
```

C.

```
3
```