Problem Set 08 · While Loops

Instructions

1. Use the answer sheet provided in the Assignment Files to complete this problem set. Fill out the header information. Follow any additional instructions that appear in the answer sheet. Submit your finished answer sheet with the rest of your deliverables.
2. You will need your PS07_academic_integrity function for this assignment. You should make any fixes necessary to make it work.
3. Read each problem carefully before starting your work. You are responsible for following all instructions within each problem. Remember that all code submissions must follow the course programming standards.
4. Below are the expected deliverables for each problem.
   - Name your files to match the format in the table below.
   - Publish your code for each problem. See PS06 for more information.
   - Do not forget to include any data files loaded into your code.

Item	Type	Deliverable to include in Submission
Problem 1: Taylor Series for cos(x)	Paired	☐ PS08_taylor_*login1*_*login2*.m ☐ PS08_taylor_*login1*_*login2*_report.pdf
Problem 2: Infinite Fin Model	Individual	☐ PS08_fin_length_*yourlogin*.m ☐ PS08_fin_length_*yourlogin*_report.pdf
Problem 3: Approximation of ln(3)	Paired	☐ PS08_ln3_approx_*yourlogin1*_*yourlogin2*.m ☐ PS08_ln3_approx _*yourlogin1*_*yourlogin2*_report.pdf ☐ Test Cases (submitted in Answer Sheet) ☐ Tracking Table (submitted in Answer Sheet)
Problem 4: No-Loop Approximation of ln(3)	Individual	☐ PS08_ln3_noloop_*yourlogin*.m ☐ PS08_ln3_noloop_*yourlogin*_report.pdf
PS08 Answer Sheet	Individual	☐ PS08_AnswerSheet_*yourlogin*.docx
Academic Integrity Statement	Individual	☐ PS07_academic_integrity_*yourlogin*.m

5. Save all files to your Purdue career account in a folder specific to PS08.
6. When you are ready to submit your assignment,
   - Compress all the deliverables into one zip file and name it **PS08_yourlogin.zip**. Be sure that you
     i. Only compress files using **.zip** format. No other compression format will be accepted.
     ii. Only include deliverables. Do **not** include the problem document, blank templates, etc.
   - Submit the zip file to the Blackboard drop box for PS08 before the due date.
7. After grades are released for this assignment, access your feedback via the assignment rubric in the My Grades section of Blackboard.

Helpful MATLAB Commands

Learn about the following built-in MATLAB commands, which might be useful in your solutions:

`round, factorial, any, all`

Problem 1: Taylor Series for $\cos x$

Paired

Learning Objectives

Your work on this problem may be assessed using any of the following learning objectives:

- Perform and evaluate algebraic and trigonometric operations
- Assign and manage variables
- Manage text output
- Perform and evaluate relational and logical operations
- Create and execute a user-defined function
- Construct and troubleshoot a flowchart using standard symbols and pseudocode
- Track a flowchart with a definite looping structure
- Create test cases to evaluate a flowchart
- Construct a flowchart using standard symbols and pseudocode
- Create and troubleshoot a selection structure
- Convert between these indefinite looping structure representations: English, a flowchart, and code
- Code an indefinite looping structure
- Track execution of an indefinite looping structure using a variable tracking table

Problem Setup

Many complicated functions used in science and engineering applications are made easier to understand when represented as Taylor series. Taylor series are also used in science and engineering applications to make approximations. For instance, the cosine function can be approximated with the Taylor series as:

$$\cos x = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k}}{(2k)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots$$

In this problem, your task is to create a user-defined function that calculates the approximate value of the cosine of a given number by summing an unknown number of terms in the series. The number of terms will be determined by establishing a "tolerance", which is the maximum allowable value of the final computed term in the series.

For example, the table below shows the number of terms, values of each Taylor series term, and the approximate value of cosine of 2 when the tolerance is 0.01.

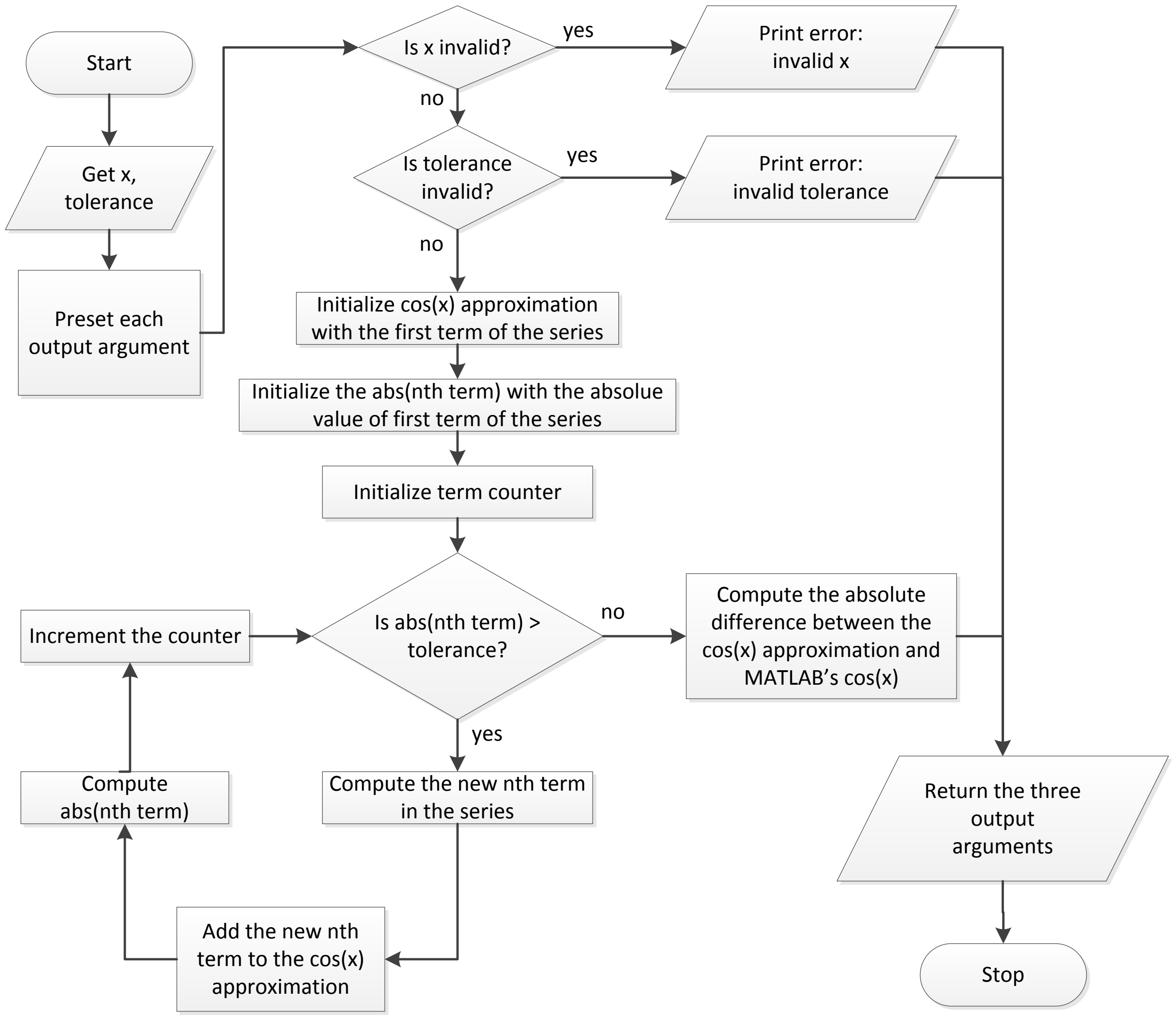
Number of Terms , *n*	Value of (*k*)	Value of *nth* Term	Taylor Series Value of cos (2)
1	0	1	1
2	1	-2	-1
3	2	0.6667	-0.3333
4	3	-0.0889	-0.4222
5	4	0.0063 (|0.0063| ≤ 0.01, so final computed term)	-0.4159
	MATLAB's built-in cosine function: cos(2) = -0.4161		

After determining the approximate value of the cosine function, your UDF needs to calculate the absolute difference between that value and MATLAB's value for the cosine. In the table above, the absolute difference is |(-0.4159) – (-0.4161)|.

Your user-defined function must:

- accept as input arguments x and the tolerance value for the last term of the Taylor series
- test for invalid inputs
    - x must be a scalar;
    - the tolerance must be between 0 and 1, not inclusive
    - print appropriate, helpful error messages for invalid inputs
- return as outputs:
    - the number of terms used in the Taylor series computation,
    - the computed value of $\cos x$ using the Taylor series, and
    - the absolute difference between the Taylor series computed value of $\cos x$ and the value returned by MATLAB's built-in cosine function
    - if an input argument is invalid, then return each output argument above as -99

A flowchart is provided for this problem and shows a method to code the Taylor series of the cosine function. Translate this specific flowchart to MATLAB code.


Start
Get x, tolerance
Preset each output argument
Is x invalid?
yes
Print error: invalid x
no
Is tolerance invalid?
yes
Print error: invalid tolerance
no
Initialize cos(x) approximation with the first term of the series
Initialize the abs(nth term) with the absolue value of first term of the series
Initialize term counter
Is abs(nth term) > tolerance?
no
Compute the absolute difference between the cos(x) approximation and MATLAB's cos(x)
yes
Compute the new nth term in the series
Add the new nth term to the cos(x) approximation
Compute abs(nth term)
Increment the counter
Return the three output arguments
Stop

Problem Steps

1. **Before you start to code**: Review the flowchart to understand the process for using the Taylor series to compute $\cos x$. Note that error messages are printed to the MATLAB Command Window.
2. In your Word answer sheet:
    a. Add a series of test cases to thoroughly test all the possible paths (valid and invalid paths) in the flowchart. One test case has been provided on your answer sheet.
    b. Record the corresponding flowchart outputs for each test case.
    c. Complete the variable tracking table by hand for the execution of the loop for the test case provided.
3. Translate the flowchart above to a MATLAB user-defined function named **PS08_taylor_cos_*yourlogin1_yourlogin2*.m**. Comment your code appropriately and follow the ENGR132 Programming Standards.

   Hint: learn about MATLAB's `factorial` command

4. Test your function by calling it from the Command Window with the test cases you created in step 2a. Do not suppress the output when you call your function. Paste the function call and results displayed in the Command Window as comments under the `COMMAND WINDOW OUTPUTS` section of your function file.
5. Call your academic integrity function in the `ACADEMIC INTEGRITY` section.
6. Publish your function as a PDF file using any valid test case and name the file as directed in the deliverables list.

Problem 2: Infinite Fin Model

Individual

Learning Objectives

Your work on this problem may be assessed using any of the following learning objectives:

- Perform and evaluate algebraic and trigonometric operations
- Assign and manage variables
- Manage text output
- Perform and evaluate relational and logical operations
- Create and execute a user-defined function
- Construct a flowchart for an indefinite looping structure using standard symbols and pseudocode
- Track a flowchart with an indefinite looping structure
- Create test cases to evaluate a flowchart
- Construct a flowchart using standard symbols and pseudocode
- Create and troubleshoot a selection structure
- Convert between these indefinite looping structure representations: English, a flowchart, and code
- Code an indefinite looping structure

Problem Setup

Electronics generate a lot of heat while operating. Computers have cooling systems embedded in their motherboards to dissipate the heat. One method of cooling is to use extended surfaces, commonly referred to as fins. The fins are a means to enhance heat transfer between the motherboard and the surrounding air by adding to the surface area over which convection cooling occurs. You can approximate the temperature along the fin using a technique called an infinite fin model. This method allows you to simplify the temperature calculations by assuming that the temperature of the fin at the end farthest away from the heat source is equal to the ambient air temperature.

The equation for the infinite fin model is

$$T = T_\infty + (T_b - T_\infty)e^{-mx}$$

where T is the temperature of the rod at a given length from the heat source, T_b is the temperature at the end of the rod that is attached to the heatsource, T_∞ is the ambient air temperature, x is the distance from the heat source along the rod, and m is a constant associated with the rod and is defined as

$$m = \sqrt{\frac{hP}{kA}}$$

where h is the heat transfer coefficient, P is the perimeter around the cross-sectional area of the rod, A is the cross-sectional area of the rod, and k is the thermal conductivity of the rod. You can assume constant thermal conductivity for this method.

You are testing fins of different materials. You plan to use rods, each 0.005 m in diameter, of copper, aluminium, and stainless steel. One end of the rods will be maintained at 373 K. The surface of the rod is exposed to ambient air at 298 K, with a convection heat transfer coefficient of 100 W/m^{2}K. The thermal conductivity of the three metals are as follows:

Metal	**Thermal Conductivity (W/(m·K))**
Aluminium	205
Copper	400
Stainless Steel	16

Your task is to determine the minimum rod length, to the nearest centimeter, necessary to use the infinite fin model. You must create a user-defined function that will calculate the temperature along the rod. This function must

- Accept four (4) valid inputs:
    - Rod diameter,
    - the metal's thermal conductivity,
    - the heat source temperature, and
    - the ambient air temperature.
- Return one output: the minimum length of the rod.
- Check that all inputs are greater than or equal to zero and prints an appropriate and useful warning message. If one of the inputs is invalid, then return a rod length of -1.
- Round the modelled temperature, T, to the nearest whole number to be in line with the accuracy of your temperature measuring device.
- Display the minimum rod length to the Command Window using professional formatting.

Problem Steps

1. **Before you start to code**: Create a flowchart to outline how information should move through the code. You can draw the flowchart using any means that result in a clear image for the answer sheet. Make sure your flowchart is legible. Options include:
    - Drawing it by hand and taking a clear photo
    - Drawing it directly in the Word answer sheet using Word's drawing tools
    - Drawing it in Microsoft's Powerpoint, Publisher, or Visio
    - Using another flowchart tool, such as Lucidchart
2. In your Word answer sheet, complete the variable tracking table by hand for the test case provided.
3. Translate your flowchart to a MATLAB user-defined function. Comment your code appropriately and follow the ENGR132 Programming Standards.
4. Test your function by calling it for a copper rod and then an aluminium rod.
5. For the two tests, paste the function call and results displayed in the Command Window as comments under the `COMMAND WINDOW OUTPUTS` section of your function file.
6. Call your academic integrity function in the `ACADEMIC INTEGRITY` section
7. Publish your function as a PDF file using stainless steel as the inputs and name the file as directed in the deliverables list.

Problem 3: Approximation of $\ln 3$

Paired

Learning Objectives

Your work on this problem may be assessed using any of the following learning objectives:

- Perform and evaluate algebraic and trigonometric operations
- Assign and manage variables
- Manage text output
- Perform and evaluate relational and logical operations
- Create and execute a user-defined function
- Track a flowchart with an definite looping structure
- Create test cases to evaluate a flowchart
- Create and troubleshoot a selection structure
- Convert between these definite looping structure representations: English, a flowchart, and code
- Code an definite looping structure
- Track execution of a definite looping structure using a variable tracking table

Problem Setup

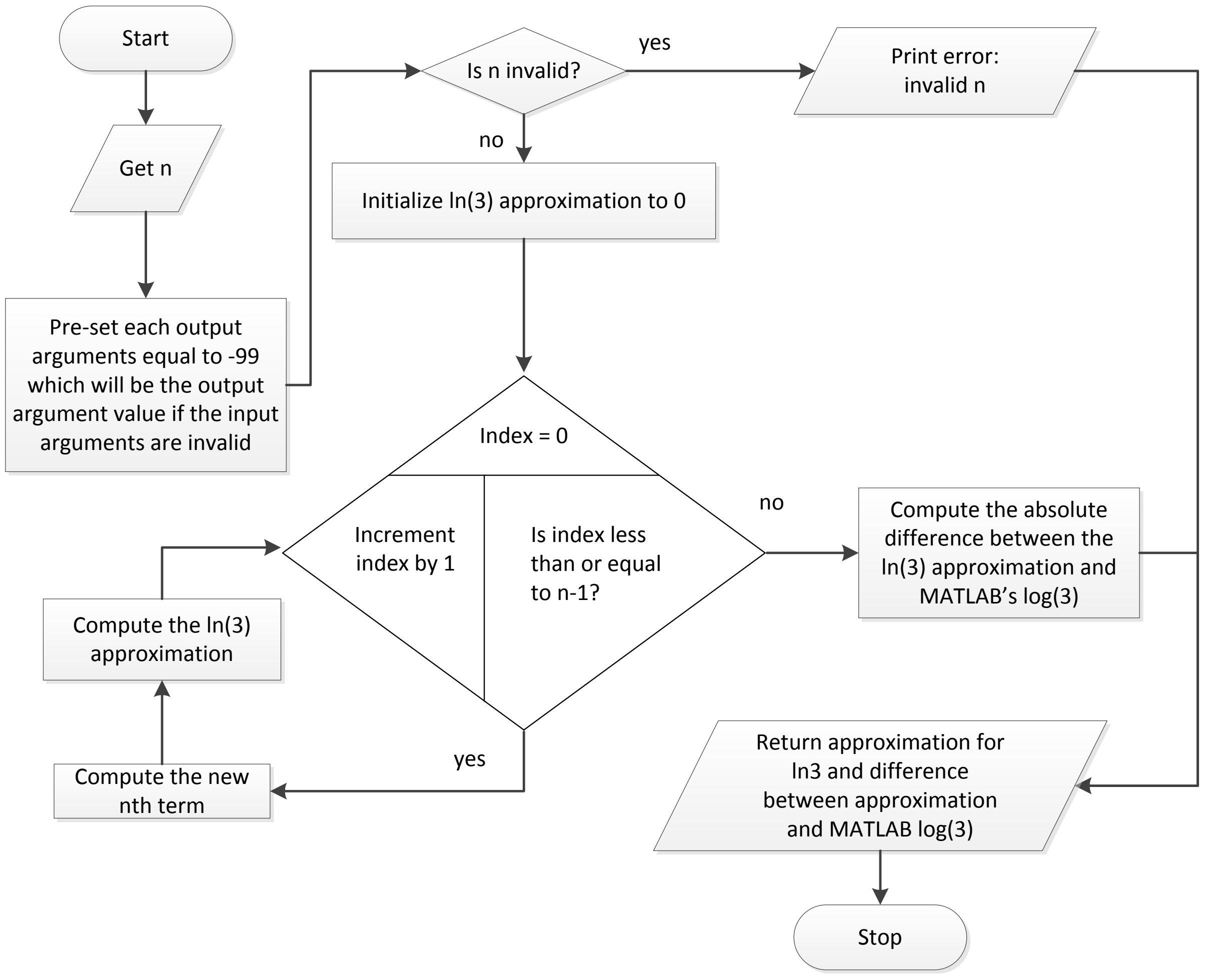
The value of $\ln(3)$ can be approximated using the following expression:

$$\ln 3 = \sum_{k=0}^{\infty} \frac{1}{4^k}\left(\frac{1}{2k+1}\right) \quad for\ k\ =\ 0, 1, 2, \ldots$$

Your task is to create a user-defined function to compute the value of $\ln(3)$ for a given number of terms. Your user-defined function must:

- Accept number of terms (n) as a scalar input argument
- test for invalid inputs
  - n must be a positive integer
  - print appropriate, helpful warning messages for invalid inputs
- return as outputs:
  - the estimate for $\ln(3)$,
  - the absolute difference between the estimate for $\ln(3)$ and the value returned by `log(3)` in MATLAB

An approved flowchart is provided for this problem and shows a method to code the approximation. Translate this specific flowchart to MATLAB code. Pay careful attention to the condition in the flowchart: the value of n is not equal to the value of k in the summation.


Start
Get n
Pre-set each output arguments equal to -99 which will be the output argument value if the input arguments are invalid
Is n invalid?
yes
Print error: invalid n
no
Initialize ln(3) approximation to 0
Index = 0
Increment index by 1
Is index less than or equal to n-1?
no
Compute the absolute difference between the ln(3) approximation and MATLAB's log(3)
Compute the ln(3) approximation
Compute the new nth term
yes
Return approximation for ln3 and difference between approximation and MATLAB log(3)
Stop

Problem Steps

1. **Before you start to code**: Review the flowchart to understand the process for using the expansion to compute $\ln 3$. Note that error messages are printed to the MATLAB Command Window.

2. In your Word answer sheet:

    a. Add a series of test cases to thoroughly test all the possible paths (valid and invalid paths) in the flowchart. One test case has been provided on your answer sheet.

    b. Record the corresponding flowchart outputs for each test case.

    c. Complete the variable tracking table by hand for the execution of the loop for the test case provided.

3. Translate the flowchart above to a MATLAB user-defined function named **PS08_ln3_approx_*yourlogin1_yourlogin2*.m**. Comment your code appropriately and follow the ENGR132 Programming Standards.

    Hint: To see variable values to more decimal places type `format long` in MATLAB Command Window.
    https://www.mathworks.com/help/matlab/matlab_env/format-output.html?s_tid=gn_loc_drop

4. Test your function by calling it with the test cases you created in step 2a. Add the following test cases:

    a. n = 5

   b. n = 10

   c. n = 20

   Do not suppress the output when you call your function. Paste the function call and results displayed in the Command Window as comments under the `COMMAND WINDOW OUTPUTS` section of your function file.

5. Call your academic integrity function in the `ACADEMIC INTEGRITY` section.
6. Publish your function as a PDF file using any valid test case and name the file as directed in the deliverables list.

Problem 4: No-Loop Approximation of $\ln 3$

Individual

Learning Objectives

Your work on this problem may be assessed using any of the following learning objectives:

- Perform and evaluate algebraic and trigonometric operations
- Assign and manage variables
- Manipulate arrays (vectors or matrices)
- Manage text output
- Perform and evaluate relational and logical operations
- Create and execute a user-defined function
- Construct and troubleshoot a flowchart using standard symbols and pseudocode
- Create and troubleshoot a selection structure
- Eliminate unnecessary definite looping structures

Problem Setup

In Problem 3, you had to follow a flowchart that outlined a method for approximating $\ln 3$ using a summation. Review that problem and your code for it.

Your task for this problem is to perform the same summation without using a loop. Your new code will be a UDF that

- Has the same inputs and outputs as defined in Problem 3
- Continues to check for invalid inputs and print error messages as defined in Problem 3
- Finds the sum of the terms for the approximate value of $\ln 3$ without using a loop.

Problem Steps

1. Using PS08_ln3_noloop_template.m file, create a function has the same functionality as Problem 3 but without a loop. Name this new function according to the naming format in the Deliverables List.
2. Test your function by calling it from the Command Window with the following test cases:
   - n = 8
   - n = 12
   - n = 24
   - n = -0.25

   Do not suppress your function call in the Command Window. This will allow the output arguments to be displayed to the Command Window. Paste the function call and results displayed in the Command Window as comments under the `COMMAND WINDOW OUTPUTS` section of your function file.
3. Call your academic integrity function in the `ACADEMIC INTEGRITY` section.
4. Publish your function as a PDF file using <u>any valid test case</u> and name the file as directed in the deliverables list.