Problem Set 10
While Loops and For Loops

Instructions

1. This problem set contains paired programming and individual programming problems. Each problem has a set of deliverables that needs to be submitted. You are responsible for following the appropriate guidelines and instructions below. Create appropriately-named files as instructed.
2. Save all files to your Purdue career account in a folder specific to PS10.
3. Compress all deliverables into one zip file named **PS10_yourlogin.zip**. Submit the zip file to the Blackboard drop box for PS10 before the due date.

Deliverables List

Item	Type	Deliverable
Problem 1: Taylor Series for $\ln x$	Paired	Test Cases (submitted on Answer Sheet) Tracking Table (submitted on Answer Sheet) PS10_taylor_ln_*yourlogin1*_*yourlogin2*.m PS10_taylor_ln_*yourlogin1*_*yourlogin2*_report.pdf
Problem 2: Approximation of $\sqrt{2}$	Paired	Test Cases (submitted on Answer Sheet) Tracking Table (submitted on Answer Sheet) PS10_sqrt2_ *yourlogin1*_*yourlogin2*.m PS10_sqrt2_ *yourlogin1*_*yourlogin2*_report.pdf
Problem 3: Medication Infusion Therapy	Individual	Flowchart (submitted in Answer Sheet) PS10_infusion_*yourlogin*.m PS10_infusion_*yourlogin*_report.pdf
Answer Sheet	--	PS10_Answer_Sheet_*yourlogin*.docx

Answer Sheet

You must place your flowcharts and test cases in the Answer Sheet provided in the Assignment Files. The answer sheet is named PS10_answer_sheet_template.docx. You must resave it as **PS10_answer_sheet_*yourlogin*.docx** before submitting it.

- Complete the assignment information at the top of the answer sheet
- Your answer sheet will be used for all problems in this set
- Follow any additional instructions on the Answer Sheet
- Place items in the correct locations on the Answer Sheet

Problem Set 10
While Loops and For Loops

Problem 1: Taylor Series for $\ln x$

Paired Programming

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, when $0 < x \leq 2$, the natural logarithm function can be approximated with the Taylor series as:

$$\ln x = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}(x-1)^n}{n} = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \cdots$$

In this problem, your task is to create a user-defined function that calculates the approximate value of the natural log 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 absolute 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 sum of the series’ terms for the natural log of 2 when the tolerance is 0.1.

Number of Terms (n)	Value of *nth* Term	Taylor Series Value of $\ln(2)$
1	1	1
2	-0.5	0.5
3	0.3333	0.8333
4	-0.25	0.5833
5	0.2	0.7833
6	-0.1667	0.6167
7	0.1429	0.7596
8	-0.125	0.6346
9	0.1111	0.7457
10	-0.1 (|-0.1| <=0.1, final computed term)	0.6457
MATLAB’s built-in natural log function: `log(2) = 0.6931`		

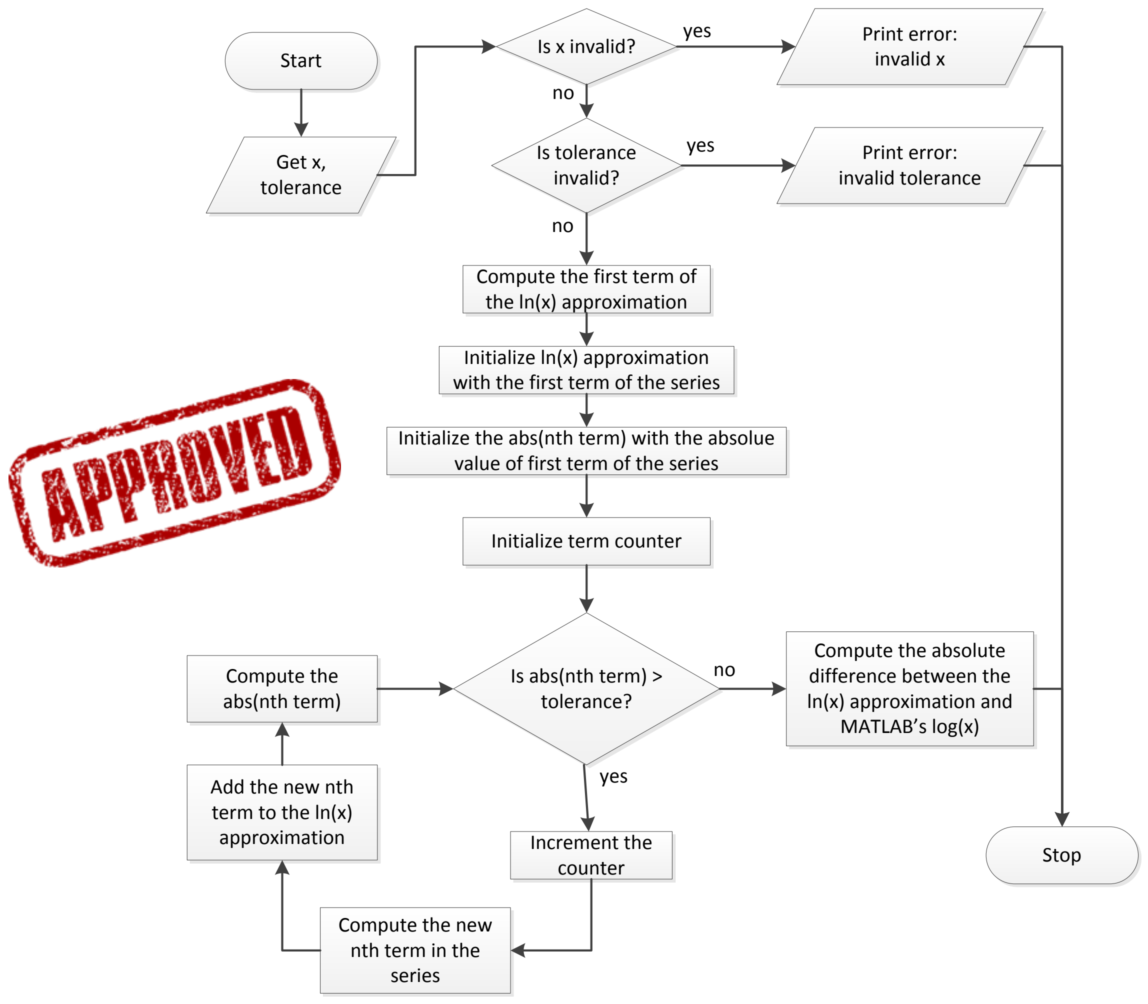
After determining the approximate value of the natural log, your UDF needs to calculate the absolute difference between that value and MATLAB’s value for the natural log. In the table above, the absolute difference is |0.6457 – 0.6931|.

Problem Set 10
While Loops and For Loops

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 $0 < x \le 2$;
    - the maximum value of the last term 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 $\ln x$ using the Taylor series, and
    - the absolute difference between the $\ln x$ approximation and the value returned by MATLAB's built-in natural log function

An approved flowchart is provided for this problem and shows a method to code the Taylor series of the natural log function. Translate this specific flowchart to MATLAB code.


Start
Get x, tolerance
Is x invalid?
yes
Print error: invalid x
no
Is tolerance invalid?
yes
Print error: invalid tolerance
no
Compute the first term of the ln(x) approximation
Initialize ln(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
APPROVED
Compute the abs(nth term)
Is abs(nth term) > tolerance?
no
Compute the absolute difference between the ln(x) approximation and MATLAB's log(x)
yes
Add the new nth term to the ln(x) approximation
Increment the counter
Stop
Compute the new nth term in the series

Problem Set 10
While Loops and For Loops

Problem Steps

1. **Before you start to code**: Review the flowchart to understand the process for using the Taylor series to compute $\ln x$. 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 **PS10_taylor_ln_*yourlogin1_yourlogin2*.m**. You must demonstrate an ability to translate the flowchart as provided. Comment your code appropriately and follow the ENGR132 Programming Standards.
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. Publish your function as a PDF file using any valid test case and name the file **PS10_taylor_ln_*yourlogin1_yourlogin2*_report.pdf**

Problem Set 10
While Loops and For Loops

Problem 2: Approximation of $\sqrt{2}$

Paired Programming

Problem Setup

The value of $\sqrt{2}$ can be approximated using the following expression:

$$\sqrt{2} \approx \sum_{k=0}^{\infty} \frac{(2k+1)!}{2^{3k+1}(k!)^2} \quad for\ k = 0, 1, 2, \ldots$$

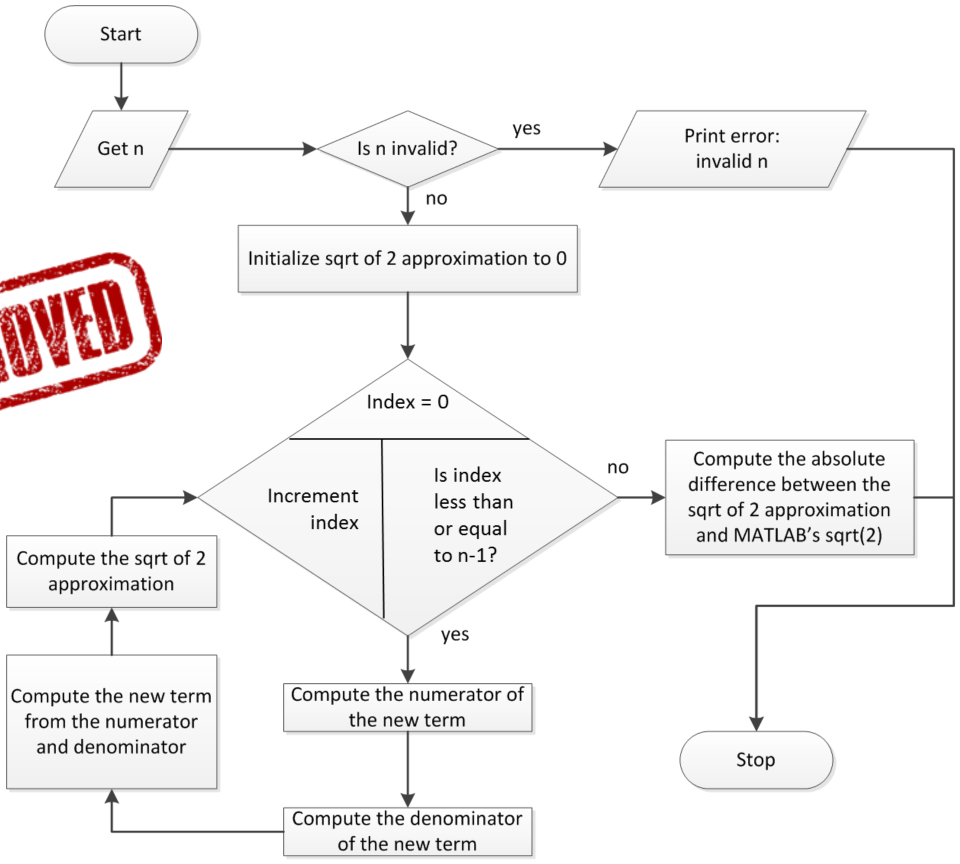
Your task is to create a user-defined function to compute the value of $\sqrt{2}$ for a given number of terms.

Your user-defined function must:

- accept the number of term (n) as a scalar input argument
- test for invalid inputs
  - n must be a positive integer
  - print appropriate, helpful error messages for invalid inputs
- return as outputs:
  - the estimate for $\sqrt{2}$,
  - the absolute difference between the estimate for $\sqrt{2}$ and the value returned by `sqrt(2)` in MATLAB

An approved flowchart is provided for this problem and shows a method to code the series approximation of $\sqrt{2}$. 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.

Problem Set 10
While Loops and For Loops


Start
Get n
Is n invalid?
yes
Print error: invalid n
no
Initialize sqrt of 2 approximation to 0
Index = 0
Increment index
Is index less than or equal to n-1?
no
Compute the absolute difference between the sqrt of 2 approximation and MATLAB's sqrt(2)
yes
Compute the numerator of the new term
Compute the denominator of the new term
Compute the new term from the numerator and denominator
Compute the sqrt of 2 approximation
Stop

APPROVED

Problem Steps

1. Before you start to code:
   a. Review the flowchart to understand the process for approximating of $\sqrt{2}$. Note that error messages are printed to the MATLAB Command Window.
2. In the Word answer sheet:
   a. Add a series of test cases to thoroughly test all the possible paths in the flowchart (valid and invalid paths). One test case has been provided.
   b. Record the corresponding flowchart outputs.
   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 **PS10_sqrt2_*yourlogin1_yourlogin2*.m**. You must demonstrate an ability to translate the flowchart as provided. Comment your code appropriately and follow the ENGR132 Programming Standards.

   Hint: learn about MATLAB's `factorial` command.

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 from the Command Window with the test cases you created in step 2a. Add the following cases:
   a. n = 10
   b. n = 25
   c. n = 50

   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. Publish your function as a PDF file using any valid test case and name the file **PS10_sqrt2_*yourlogin1*_*yourlogin2*_report.pdf**

Problem Set 10
While Loops and For Loops

Problem 3: Medication Infusion Therapy

Individual Programming

Problem Setup

A pharmaceutical company has asked your engineering firm to build a prototype infusion machine that will be used with its newly released medication that is administered to a patient as a two-phase intravenous infusion.

As part of that project, you have been asked to create a user-defined function that will calculate how long an infusion will last given a patient's weight and medication dosage.

The medication is prescribed with units of milligram of medication per kilogram of patient weight (mg/kg). It can be prescribed in doses 25-100 mg/kg and cannot be given to anyone who weighs less than 40 kg or more than 175 kg. A sample patient who weighs 100 kg and is prescribed 80 mg/kg would need to receive 8,000 mg of medication by the end of an infusion.

The first phase of an infusion is a titration in which the infusion rate increases each minute. Every patient receiving the infusion will go through this phase. In the first minute, the infusion rate is equal to 75 (mg/min). After that, the rate increases by 50% each minute, rounded to the nearest integer. The number of rate increases is found by rounding .05 times the patient weight to the nearest integer. An example is shown in Table 1.

Table 1: First infusion phase for a 100-kg patient, who requires five rate increases from the initial rate

Infusion rate (mg/min)	75	113	170	255	383	575
Total administered dose at the end of the elapsed time (mg)	75	188	358	613	996	1571
Elapsed infusion time (min)	1	2	3	4	5	6

The sample patient in table has received 1571 mg of their 8000 mg dose at the end of the first phase, which took 6 minutes to complete.

In the second phase, the infusion rate increases by 10% every minute until the remainder of the medication has been administered. Note that the total administered dose at the end of the final minute may be slightly greater than the prescribed dose; that is allowed for this problem.

Problem Set 10
While Loops and For Loops

Table 2: Second infusion phase for the same sample patient in Table 1.

Infusion rate (mg/min)	633	696	766	…	final infusion rate
Total administered dose (mg)	2204	2900	3666	…	total administered dose
Elapsed infusion time (min)	7	8	9	…	final time

Your user-defined function must

- accept a patient's weight in kilograms and medication dose in mg/kg,
- determine the total time for the infusion (minutes) and total administered dose (mg), and
- return the total infusion time and administered dose.

Problem Steps

1. **Before you start to code**: Create a flowchart to outline how information should move through the code.
   - Open the Answer Sheet and examine the high-level flowchart.
   - Flowchart the checks for valid inputs validity tests, the first infusion phase, and the second infusion phase.
   - Draw the flowchart sections using any means that result in a clear image for the answer sheet. Make sure your flowchart is legible!
2. In your Word answer sheet:
   a. Insert a clear image for each of your flowchart sections.
   b. Complete the variable tracking tables for the first and second phases using the test case provided.
3. Translate your flowchart into a two-input, two-output user-defined function named **PS10_infusion_*yourlogin*.m**.
4. Test your function by calling it from the Command Window with the test case used in step 2b. 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.
5. Publish your function as a PDF file using test case 2b above and name the file **PS10_infusion_*yourlogin*_report.pdf**

Image reference: http://wasatchinfusion.com/services/ivig/