Before beginning this homework, create a new Notepad++, **H5_LastName.txt**, file in your cs7sXX MyDocuments folder. This homework is to be completed individually.

**PART ONE: SIMPLE FUNCTIONS AND LOOPS**

1. Let’s do some work with **for-loops**. For-loops are a form of iteration that allow you to perform the same instructions a fixed number of times. The instructions that get performed are those in the **loop body**. In MATLAB, the loop body is the part that is typically indented and between the keywords `for` and `end`.

   ```matlab
   for loopvar = range
       loop body statement(s)
   end
   ```

2. For example, enter the following code into the Command Window:
   ```matlab
   >> prod = 1;
   for k = 1:5
       prod = prod * k
   end
   ```

3. The above code will multiply up the numbers 1 through 5 and store it in a variable `prod`. During each iteration, `prod` is being updated so that its new value is equal to its old value multiplied by the loop variable `k`.

   - At the beginning of the first iteration, the loop variable `k` will begin at the **start** of the range specified. In this case, `k` begins at 1.
   - At the beginning of the second iteration, `k` is incremented by the **step size** specified. If the step size is not specified, `k` will increment by 1. In this case, `k` equals 2 on the second iteration since no step size is specified.
   - After every iteration, the loop variable will continue to change by the step size until its value is outside of the end of the range specified.

4. The loop body, which executes during every iteration of the loop, updates the value of the variable `prod`. The old value of `prod` is multiplied by the loop variable `k` and this product becomes the new value of `prod`.

   ```matlab
   prod = prod * k
   ```

   The right side of the assignment operator `=` is evaluated first, and then the result is assigned to the variable on the left side of the assignment operator. For this reason, the variable `prod` shows up on both sides of the assignment.

**QUESTION #1.** In this for-loop, the variable `k` serves 2 important purposes. What are they? (Note that the loop variable in the next section of this homework will function similarly).
5. Notice that we didn’t put a semicolon at the end of the line in the loop body. This is NOT normal and is only for demonstration purposes. Usually, you DO NOT want to print meaningless information out during every iteration of the loop.

**QUESTION #2.** Why might you not want to print out information in a loop body? **HINT:** Think about loops that go from 1 to 1000 or 1 million.

6. Let’s go ahead and turn this into a more general function. Create a new function called `multiply_nums`. This function will have two inputs (start and finish), and one output prod which is the total number you get when you multiply up the integers between start and finish.

7. Add comments at the top of your function.

**QUESTION #3.** Call your function so that it calculates the product of all the integers between 1 and 20. Write the answer AND the function call.

**PART TWO: FUNCTIONS, LOOPS AND SEARCHING FOR VALUES AND LOOPS**

1. Now you will write a function that will output a student’s grade by calculating their average score between two tests. This is the table of test scores that you will be looking up students in:

<table>
<thead>
<tr>
<th>Student ID</th>
<th>Test#1 Score</th>
<th>Test#2 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1542</td>
<td>73</td>
<td>79</td>
</tr>
<tr>
<td>2084</td>
<td>86</td>
<td>65</td>
</tr>
<tr>
<td>1541</td>
<td>90</td>
<td>80</td>
</tr>
<tr>
<td>3454</td>
<td>85</td>
<td>60</td>
</tr>
<tr>
<td>3452</td>
<td>70</td>
<td>62</td>
</tr>
</tbody>
</table>

2. Create the variable `mat` in the Command Window. *This variable is a matrix representing the table of test scores above.*

```matlab
>> mat=[1542, 73, 79; 2084, 86, 65; 1541, 90, 80; 3454, 85, 60; 3452, 70, 62];
```

*You will need to keep this matrix in the Workspace in order to use it as input into the function.* `mat` will be used by the function to find the row that corresponds to the student whose grade you would like to calculate.

3. The function will be called `calcGrade`. The first input `studentID` is the ID number of the student that you would like to calculate. The second input `scores` is a matrix of ID numbers and scores (i.e. `mat`). The function should then output `grade` which will be the average of the two test scores.

4. Before writing the function body, let’s consider some test cases. This is common practice in software engineering, and thinking about the tests before writing the function allows us to really understand the function’s behavior.

**QUESTION #3.** What should the function return if you call it with the following inputs (2084, mat)?
QUESTION #4. Your function should only calculate a grade for students that exist in the score sheet. What would you want to do if the student ID passed into `calcGrade` does not exist in the score sheet? For example, what do you think the function should return if you call it with `(5678, mat)`? (You don’t have to implement this, just describe how you would handle this sort of input within your function).

QUESTION #5. Your function should only calculate a grade for students that took both tests. What would you want to do if the score sheet passed into `calcGrade` contains a student that has not taken both tests? (You don’t have to implement this, just describe how you would handle this sort of input within your function).

5. Now let’s go ahead and write the function `calcGrade`. The following steps (#6-#10) will guide you through the implementation but this is a basic skeleton of what the function should look like:

```matlab
<INSERT FUNCTION HEADER HERE>
% Add Comments
for j = 1: <HOW MANY TIMES DO YOU LOOP?>
    <INSERT FUNCTION BODY HERE>
end
end
```

6. Write the function header (if you haven’t already) before moving to the next step. Refer to step 3 for help.

QUESTION #6. What is the built-in MATLAB function that gives us the number of rows in a matrix?

QUESTION #7. In what situations would you want to call the function `length` instead of `size`?

7. Now in the function body, you will use the loop variable `j` to iterate through every row of the first column of `scores`. Remember that `scores` is the input variable that exists within the `calcGrade` function and `mat` is the value being passed into the function. Based off this information, determine the end range for the loop variable `j`. In other words, how many times do you loop?

8. At every row, you will check if the student ID matches the `studentID` (the input) that you want to find. Notice that the column remains the same during every iteration of the loop, but the row is changing.

   - On the 1st iteration of the loop (`j = 1`), check the first row and the first column of `scores`. Below is the `scores` table which is currently holding the values of `mat` from step 1.

<table>
<thead>
<tr>
<th>Score 1</th>
<th>Score 2</th>
<th>Score 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1542</td>
<td>73</td>
<td>79</td>
</tr>
<tr>
<td>2084</td>
<td>86</td>
<td>65</td>
</tr>
<tr>
<td>1541</td>
<td>90</td>
<td>80</td>
</tr>
<tr>
<td>3454</td>
<td>85</td>
<td>60</td>
</tr>
<tr>
<td>3452</td>
<td>70</td>
<td>62</td>
</tr>
</tbody>
</table>

   - On the 2nd iteration of the loop (`j=2`), the loop variable `j` is increased by one.
Check the second row and the first column of scores.

(j,2)

1542  73  79
2084  86  65
1541  90  80
3454  85  60
3452  70  62

**QUESTION #8.** What is the equality comparison operator in MATLAB?

9. On every iteration of the loop, use an **if-statement** to check if value in the \( j \)th row and first column of scores matches value of the input variable `studentID`.

10. If the two values are equal (the condition is true), calculate the total grade using the two test scores that correspond to the student. This means that you must grab the value in the second column (\( \text{score1} \)) and the value in the third column (\( \text{score2} \)). Calculate the average grade by adding together the two test scores (\( \text{score1} \) and \( \text{score2} \)) and dividing by 2. The result of this calculation must be assigned to the output variable `grade` in order to for it to be returned from the function.

**QUESTION #9.** Verify that the function is working. Call the finished function `calcGrade` using the inputs from **QUESTION #3**. Check that the function returned the grade you were expecting. Write the answer AND the function call.

**PART THREE: APPENDING TO A MATRIX**

1. For next week's lab, you will need to know how to build a matrix up by iterating through a loop, so let's go ahead and learn that now. Writing a function for this is not necessary, so instead you will write a **script** called `build.m`.

2. Building a matrix implies adding either a new row or a new column to it.

3. Start with the matrix `my_mat = [1, 2, 3, 4, 5; 2:6; 3:7];`

**QUESTION #10.** What are the dimensions of the variable `my_mat` (i.e. # of rows by # of columns)?

4. Now add 8 new rows to `my_mat` using a for-loop such that the values follow the same pattern as the previous rows. Each new row must have the same number of columns as the original `my_mat` matrix. Type the following lines of code into your script.

```matlab
for row_count = 4:11
    my_mat(row_count, :) = [row_count, row_count + 1, row_count + 2, row_count + 3, row_count + 4]
end
```

**HINT:** The following table is meant to help you visualize what `my_mat` looks like after this for loop is run. See if you can draw what `my_mat` looked like before the for loop above was run and compare it to the table below!

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
5. Notice that each time through the loop, you are growing the `my_mat` matrix (there are ways to improve MATLAB’s performance with this later, but for now this works just fine).

6. The right hand side of the assignment creates a 1x5 vector that gets appended to the bottom of the `my_mat` matrix.

7. Your Turn. Clear your workspace and **comment out the above loop**.

8. Create a loop in your script that works just like the above loop ONLY this time it appends a new column to the `my_mat` matrix. Remember that you can change the name of your loop counter variable (e.g. `column_count`).

9. Change the loop so that it adds **20 new columns**.
   
   The following table is meant to help you get started. The first appended column is shown but it is up to you to determine what the subsequent columns will look like. **Drawing out what each appended column will look like will help you determine the pattern and implement the body of the loop.**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

**QUESTION #11.**

(a) Copy and paste what the 18th column of `my_mat` looks like after you run the build script.

(a) Copy your loop body into your Notepad++ report AND save your script.

---

**PART FOUR: SAVING and SUBMITTING**

1. **Submit *FOUR* THINGS:** Save your Notepad++ file. Save a copy of your `build.m` file, your `calcGrade.m` and `multiply_nums.m` file into your cs7sXX directory.
HOMEWORK #5

● The Homework#5 assignment is due next week in YOUR lab section at the BEGINNING of Lab.
● You will be graded individually.
● Don’t forget to comment at least one of your functions/ scripts.
● You will need to show the Tutor/TA that you are able to use the MATLAB commands from the homework assignment. He/She will ask questions to determine how knowledgeable you are.
● You should have one file named “Hw5_LastName.txt”, saved in your cs7sXX MyDocuments folder. (See Homework # 1 details)

***Optional***

CHALLENGE A TUTOR: If you think you have truly mastered all the concepts until here take this challenge and present your solution to a tutor when you get a checkoff.

The fibonacci series is a mathematical series of numbers such that any number is a sum of its two preceding numbers. EX: 0..1..1..2..3..5..8..13..21..etc. Write a program such that your program prints out 3 x n rows of the fibonacci sequence:
If n = 5, you should have 5 columns of the fibonacci series

ans =
     0     2     8    13
     1     3    13    21
     1     5    21    34

If n = 7, you should have 7 columns of the fibonacci series

     0     2     8    34   144   610  2584
     1     3    13    55   233   987  4181
     1     5    21    89   377  1597  6765

We will give you the function header along with the starting matrix. It is your job to append new columns and place values into them.

```
function [fib] = fibonacci_seq(n)
    %This function will output the first n columns of the fibonacci sequence
    fib = [0 2;1 3;1 5];
```

Hints: If you have the matrix mat = [1,2,3;4,5,6;7,8,9] what does mat(3) or mat(5) mean?