In place of a signature, please include a commented statement in your code affirming your recognition of the Academic Honesty Policy for the Exam 2. You will replace the blank with your name and UHID as an acknowledgement in your starting file as acknowledgement of this policy.

```
% I,<INSERT FULL NAME> (<INSERT UHID>) acknowledge that the Exam 2 for ENGI 1331,
% is to be completed by myself with no collaboration with anyone.
% I have read the ENGI 1331 Position on Academic Honesty and agree
% to abide by its provisions while taking this exam.
% I acknowledge that my submission will be run through a similarity code.
% Any student with unacceptable levels of commonality with peers or
% other sources will be brought up for an academic honesty violation.')
```

INSTRUCTIONS:

This virtual computer exam will be given on Saturday, April 3, 2021.

- You must be logged in with video on to your virtual classroom during the entire exam. If you lose connection during the exam, re-enter the virtual classroom and proceed with the exam.
- Computer with internet access and video is required.

General Rules

- You should **suppress** all output to the Command Window except if specific formatted output is requested.
- You must use any variable names specified. **If a variable name is not specified, you may create your own name for the variable.**
- **Do Not Hard Code for a specific case – your code must be flexible based on the instructions provided.**

Exam Timeline and Submission:

You will **have 1 hour and 20 minutes to complete all tasks after the 15-minute downloading/planning period and 5 minutes to upload your code**. After time is called, you will be expected to close out of MATLAB and zip your exam folder.

- During the exam, **NO COLLABORATION** of any kind (messaging of concepts, code or sharing of files) is permitted. If you are found to have collaborated during the exam, you will be brought up on academic honesty violations with the appropriate penalty enforced.

After the exam is completed and the files are submitted, any procedure (such as opening the file) which alters the date / time stamp of the file will void any allowances for mis-saved files – in other words, **after the exam is over DO NOT OPEN the file again!**

Saving Exam File

You will be expected to have one main script file associated with your exam submission in addition to any starting files, exported files, or function files. Save your script file in the exam folder on your desktop as **Exam2_cougarnet.m**. All other functions that are requested should be named as instructed with the correct function name. You must submit a .ZIP folder named **Exam2_cougarnet.zip** that contains your main script and all supporting files.

MATLAB Programming (80 points) – Approx. 80 minutes + 15 minutes for planning

In the **Exam2Data.csv**, you are provided with a matrix of data. Following the tasks below, you will create a code that allows the user to examine the entire data set and a specific column entered by the user.

NOTE (avoid hardcoding): Your function and script should produce different results if the size of the data set or individual values change.

Tasks:

Task 1 (10 min) – 10 pts

Main Script (10 pts)

Load in **Exam2Data.csv**.

- Determine the number of values in the matrix that are below 5 and output the total to the command window.
- For each value in the matrix less than 5, replace with the average of all values greater than 5

Task 2 (20 min) – 20 pts

Main Script (20 pts)

Prompt the user to **enter** (user input) a column number. Assume the number entered is a whole number. If the number entered is less than 0, take the absolute value and produce a warning. If the number entered is greater than the number of columns in the imported data or equal to zero, produce an error statement and terminate the program. You can assume the absolute value of a negative number entered is less than or equal to the number of columns.

For the values in the column entered (x), calculate the f values (length equal to the number of rows) using the piecewise function given in Equation 1.

NOTE: Assume all values are less than or equal to 100.

$$f(x) = \begin{cases} (x * 0.1) + 10, & x \leq 10 \\ e^x, & 10 < x < 100 \\ x^{2.1}, & x = 100 \end{cases} \qquad \text{Eq. 1}$$

Export the column vector of results for *f* as a .csv file named **Task2Results.csv**. Check your results for column 20 in Task2Results_20.csv and for column 2 in Task2Results_2.csv. These additional files are in the exam folder for your reference.

Task 3 (40 min) – 50 pts

Function (20 pts)

Create a function named **Peaks_cougarnet** (replace with your cougarnet username) that starting with the 4th value of the input vector (**in1**), determines if that value is a peak by checking if it meets the BOTH of the following conditions:

- Greater than the previous three values
- Greater than one and a half times the average of the following two values

If the value is a peak, store the value and location in two separate vectors (function outputs)

Function Inputs:

1. Vector

Function Outputs:

1. Vector of Peak values
2. Vector of Peak locations

NOTE: (1) The first three and last two values should not be checked.
(2) The length of each output variable should equal the number of peaks found

The function header should be formatted similarly to the following:

```
function [out1, out2] = Peaks_cougarnet(in1)
```

Remember you are free to use whatever variable names you want, but they must be listed in the same order as given in the input/output lists provided above.

Main Script (25 pts)

For the column entered in Task 2, use **Peaks_cougarnet** to the determine the Peak value(s) and their location(s). Output to the command window a formatted table using a loop (do not use the table function) as shown in the sample output. Determine the number of peaks found and store in **NumPeaks**.

Using a menu, ask the user if they would like to repeat the program starting at Task 2 for another column.

- If the user exits out of the menu, ask the user again until they select yes or no. If after the 5th time, the user still exits out of the menu, produce a warning that the program is ending and assume the selection is no.
- If yes, repeat. For each repetition, store the number of peaks found in **NumPeaks** without overwriting the previous value. The length of this vector should equal the number of times the program is repeated, including the first run.
- If no, export **NumPeaks** as a .mat file named **Task3Peaks.mat**. Check results in sample output.

Sample Output

Test Case 1: Column entered greater than number of columns in data set

```
Command Window
  There were 45 values below 5.

  Enter a column number: 40
  Error using Exam2Sp2021_cougarnet (line 26)
  Column entered not valid. Program terminated
```

Test Case 2: Column entered is negative. Code only run once.

Refer to Task2Results_20 to compare your results for Task 2.

```
Command Window
  There were 45 values below 5.

  Enter a column number: -20
  Warning: Column entered was negative.  Assumed positive

  Value    Index
  93.0     14
  98.0     20
  92.0     27
  85.0     40
  94.0     43
fx 69.0     47>>
```

Variables - NumPeaks

NumPeaks

1x1 double

	1	2
1	6	
2		

Test Case 3: Code runs 4 times.

Refer to Task2Results_2 to compare your results for Task 2.

Command Window

```
There were 45 values below 5.

Enter a column number: 10

Value   Index
100.0   4
81.0    9
92.0    13
91.0    45

Enter a column number: 7

Value   Index
70.0    11
84.0    21
98.0    22
73.0    29
98.0    34
94.0    41

Enter a column number: 9

Value   Index
92.0    5
95.0    11
83.0    15
83.0    19
68.0    24
91.0    29
81.0    33
88.0    40
98.0    43

Enter a column number: -2
Warning: Column entered was negative.  Assumed positive

Value   Index
82.0    10
96.0    21
53.0    31
54.0    32
100.0   39
89.0    46>>
```

Variables - NumPeaks

NumPeaks

1x4 double

	1	2	3	4
1	4	6	9	6