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, October 31, 2020 (Happy Halloween!)

- 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

Programming in MATLAB (only portion): This exam should take you approximately 1 hour and 20 minutes to complete and 20 minutes for planning and 5 minutes to upload your code. Total time is 1 hours and 45 minutes.

Finishing the exam

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

- During the exam, **NO COLLABORATION**, or you will be dismissed with a **ZERO**.

After the exam is completed and the file collected, 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 script file associated with your exam submission and any starting files or exported files. Save your script file in the exam folder on your desktop as **Exam2_cougarnet.m**. All other functions requested should be the named as instructed with correct cougarnet. You must submit a .ZIP file named **Exam2_cougarnet.zip** that contains the script necessary to run your code.

MATLAB Programming (80 points) – Approx. 80 minutes + 20 minutes for planning

Problem #1 (40 pts) – SAMPLE OUTPUT ON NEXT PAGE

The Aksu-Burleson (AB) sequence is an integer sequence calculated with previous values, **m**, of the sequence, using the equation below:

$$AB_n = \frac{n}{1}(AB_{n-1}) + \frac{n}{2}(AB_{n-2}) + \cdots + \frac{n}{m}(AB_{n-m}) \quad (1)$$

where $\mathbf{AB_n}$ is the nth value in the sequence **AB**, **n** is the position in the sequence, **m** is the number of previous values and can only be 1, 2 or 3.

The AB sequence requires the first **m** number of values before it can calculate the rest of the sequence, given in the form of a vector of length **m**. Your code should accept a vector for the first **m** values and then calculate the AB sequence to some number of integers, **n**.

Sample AB sequence for an integer sequence to 7 (n = 7) looking back 3 (m = 3):

AB_1 (Given)	AB_2 (Given)	AB_3 (Given)	AB_4	AB_5	AB_6	AB_7
0	1	1	12	85	672	6146

Function (20 pts) *Recommended Time: 20 minutes*

Create a function named **AB_cougarnet** (replace with your courgarnet username) that, using the 1 x m starting vector creates a sequence based on the Eq. 1 with the total number of elements defined by **n**. REMINDER: the **m** can only be 1, 2, or 3.

Function Inputs:

1. Starting vector
2. Number of values for sequence (**n**)
3. Number of values to look back (**m**)

Function Outputs:

1. Final AB sequence

The function header should be formatted similarly to the following:

```
function [out1] = AB_cougarnet(in1,in2,in3)
```

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 (20 pts) *Recommended Time: 15 minutes*

User inputs and validations:

1. Ask the user to enter the length of sequence and store this value in a variable named **n**. No validation required.
2. Ask the user to enter the number of values to look back, and store this value in a variable name **m.**
3. Check if the value entered for **m** is 1, 2 or 3. If not, ask the user to enter the value again until an appropriate value is entered.
4. Ask the user to enter a 1 x **m** or **m** x 1 vector and store the vector in a variable named **START**.
5. Check if the length of **START** is equal to **m**. If not, do the following validation:
   - If the length of **START** is greater than **m**, remove the extra values at the end of **START** and produce a warning.
   - If the length of **START** is less than **m,** produce an error and terminate program.

Using your function, **AB_cougarnet**, determine the final AB sequence and save the result to **Exam2_AB.csv**.

Problem #1: Test Case

Test Case 1:

```
Command Window
  Enter the length of the sequence (n): 7
  Enter the number of values you are looking back (m): 3
  Enter the starting values as a vector [1 x m]: [0,1,1]
fx >>
```

AB

1x7 double

	1	2	3	4	5	6	7
1	0	1	1	12	85	672	6146

Test Case 2:

```
Command Window
  Enter the length of the sequence (n): 9
  Enter the number of values you are looking back (m): 4
  Enter the number of values you are looking back (m): -2
  Enter the number of values you are looking back (m): 0
  Enter the number of values you are looking back (m): 1
  Enter the starting values as a vector [1 x m]: [3]
fx >>
```

AB

1x9 double

	1	2	3	4	5	6	7	8	9
1	3	6	18	72	360	2160	15120	120960	1088640

Test Case 3

```
Command Window
  Enter the length of the sequence (n): 5
  Enter the number of values you are looking back (m): 2
  Enter the starting values as a vector [1 x m]: [1]
  Error using Exam2_cougarnet (line 16)
  m and the starting vector are inconsistent. Program terminated
```

Test Case 4

```
Command Window
  Enter the length of the sequence (n): 5
  Enter the number of values you are looking back (m): 2
  Enter the starting values as a vector [1 x m]: [0,2,5]
  Warning: The starting vector was cut off
```

AB

1x5 double

	1	2	3	4	5
1	0	2	6	40	260

Problem #2: Description – SAMPLE OUTPUT ON NEXT PAGE

In the **Exam2data.mat**, you are provided with a matrix of grades, **GRADES**, and the name of each student, **NAME**. You are tasked with checking a user-specified row of **GRADES**. Each row represents a different student and each column represents their grade for that week (15 columns = 15 weeks and 16 rows = 16 students).

NAME

1	Ivan
2	Zion
3	Marco
4	Christian
5	Tanner
6	Javi
7	Amie
8	Clint
9	Colby
10	George
11	Jeff
12	Joey
13	Keegan
14	Rukaiya
15	Sofia
16	Tuong

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	99	91	94	68	82	60	79	61	90	92	65	62	60	68	94
2	96	91	81	80	91	86	87	62	86	94	61	60	85	61	55
3	93	81	62	74	87	61	85	69	86	90	71	71	84	69	84
4	94	87	95	78	95	71	72	71	66	79	65	83	89	77	61
5	99	91	63	78	89	94	84	83	66	83	70	90	82	82	95
6	99	93	77	72	80	83	62	60	95	89	85	87	85	82	91
7	99	80	87	74	86	85	93	90	87	72	74	82	67	82	74
8	93	89	90	80	87	86	75	80	94	65	84	89	73	67	78
9	91	82	95	67	95	63	77	80	85	63	78	60	63	84	63
10	93	83	77	73	83	92	73	79	80	66	75	95	95	71	64
11	97	92	75	78	89	81	89	63	80	79	85	75	94	61	84
12	98	84	92	86	94	95	90	67	63	70	64	70	92	93	81
13	92	86	71	95	81	85	93	60	90	63	72	79	66	75	93
14	90	81	79	63	75	82	89	90	94	78	65	85	77	75	71
15	92	94	61	86	91	66	65	78	67	94	86	70	71	77	95
16	91	90	78	76	81	65	89	82	80	62	95	95	60	73	64

GRADES
(Column # = Week #)

Function (20 pts)

Recommended Time: 25 minutes

Create a function named **ALERT_cougarnet** (replace with your courgarnet username) that, by looking at the student's grades (row vector) from week (column) to week (column), determines if a student has a decline in their performance.

Function Inputs:

1. Vector of grades

Function Outputs:

1. Grade(s) when there is a decline
2. Week(s) of the decline in grade

The function header should be formatted similarly to the following:

```
function [out1,out2] = ALERT_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.

For a decline, **BOTH** of the criteria below must be true

(1) Current grade is lower than the average of the previous two weeks.
(2) Current grade is greater than the grade for the next week (column)

An example is below with decline grades identified in weeks 3, 11, and 14 (highlighted). The sample calculation is shown for week 11 using the previous values (purple) and the next value (red).

Week -	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
GradeVector -	96	91	81	80	91	86	87	62	86	94	61	60	85	61	55
	Do Not Check														Do Not Check

(1) 61 > ((86+94)/2)
(2) 61 > 60
Store for this check:
Grade = 61
Week = 11

Final output: out1 = [81,61,61]
out2 = [3,11,14]

When a decline is found, store the current grade (value) and associated week (column) for the decline. If multiple declines occur, the functions should store all grades and weeks when there was a decline without overwriting the previous values. If no declines occur, it should result in an empty vector for both outputs.

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

Main Script (20)

Recommended Time: 20 minutes

Load **Exam2data.mat** into your script. Using a menu, ask the user to select a student from **NAME**. This will represent the row of grades you are checking.

Using your function **ALERT_cougarnet.m**, determine the grade(s) and week(s) when of the decline for the selected row of grades.

Ask the user if they would like to repeat the program for another student.

- If yes, repeat. For each repetition, store (without overwriting the previous value) the name of the student selected in a new variable named **SELECTED** and the number of declines found in **DECLINES**. NOTE: the length of **SELECTED** and **DECLINES** should equal the number of times the program is repeated.
- If no, produce a formatted output in an aligned table as shown in the sample output (do not use table function) with the name and number of declines for all student's selected.

Sample Output for given data:

4 names selected: Javi, Zion, Jeff, Rukaiya