ENGR 121: Computation Lab I

Programming Assignment 2

This assignment comprises of three problems. It is due by 11:59 pm, December 2, 2015. You must work on this assignment on your own. Plagarized code will result in a grade of zero for the assignment.

Please read the *Submission Instructions* section towards the end of this document that describes how to package your code and submit it on-line.

1. (**10 points**) The sine function can be evaluated by the following infinite series:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

Write a function `approx_sine` that takes as input two parameters, `x` and `threshold`, to perform the following task. Starting with the simplest approximation $\sin(x) = x$, add terms one at a time to improve the approximation until

$$\left| \frac{\text{true} - \text{approximation}}{\text{true}} \right| < \text{threshold}.$$

In the above, `true` denotes the value returned by MATLAB's built-in **sin** function. Your function must return two output values: the approximate value of $\sin(x)$ and the number of terms that were needed to obtain this value subject to the desired error threshold.

The following is an example of function behavior for $x = \pi/5$.

```
>> format long
>> x = pi/5;
>> real = sin(x)
real =
0.587785252292473
>> threshold = 1e-6;
>> [approx, terms] = approx_sine(x, threshold)
approx =
0.587785210384344
terms =
3
>> threshold = 1e-9;
>> [approx, terms] = approx_sine(x, threshold)
approx =
0.587785252443038
terms =
4
```

```
>> threshold = 1e-12;
>> [approx, terms] = approx_sine(x, threshold)
approx =
0.587785252292092
terms =
5
```

Note that the `terms` value returned by the function only counts the number of terms added to the starting approximation of $\sin(x)$.

2. **(10 points)** The *mode* of a data set is the value that appears most frequently in it. The built-in function in MATLAB for this is called `mode`. For example:

```
>> x = [9 10 10 9 8 7 3 10 9 8 5 10];
>> mode(x)
ans =
     10
```

Note that if there is more than one value with the same (highest) frequency, the smaller value is returned as the mode. In the following case, as 3 and 8 appear twice in the vector, the smaller value 3 is returned as the mode:

```
>> x = [3 8 5 3 4 1 8];
>> mode(x)
and =
     3
```

Therefore, if no value appears more frequently than any other, the mode of the vector will be the same as the minimum value. Write a function `my_mode` that returns the mode of a given vector without using MATLAB's built-in function. The function takes a vector of some arbitrary length as the input argument and returns the mode as the output. For example:

```
>> vec = randi([1, 10], 1, 10)
vec =
     8     1     3     1     1     9     7     4    10     1
>> m = my_mode(vec)
m =
     1
>> vec = randi([5, 10], 1, 8)
vec =
     7     7     9     9     6     7     7     9
>> m = my_mode(vec)
m =
     7
>> vec = [1 1 1 2 2 3 3 3 4 5]
vec =
     1     1     1     2     2     3     3     3     4     5
>> m = my_mode(vec)
m =
     1
>> vec = [1 2 3 4 6 8 9]
vec =
     1     2     3     4     6     8     9
>> m = my_mode(vec)
m =
     1
```

3. **(20 points)** You are asked to develop a tiered ranking system for video game players based on their performance in a tournament.[1] Each player participated in 100 matches and is assigned a score based on the number of wins that they have had. A bonus is given to the player's score if he or she elected to have a handicap throughout the duration of the tournament. Penalties are given if the player was found to have had outside help during the tournament. The inputs to this problem are provided as:

```
clear,clc,close all        % Clear all

% Inputs Variables
N = 50;                   % Number of players considered
A = randi([0 100],1,N);     % Number of matches won
B = round(rand(1,N));       % Bonus Assigned
P = randi([0 4],1,N);       % Penalty Assigned
% Penalty Assigned (Weighted towards 0)
P(P>2)=0;
X = [A;B;P];               % Input Data
clear A B P                % Refer to only X in this problem
```

The process of creating the ranking system is made up of three parts: score adjustment, determining metrics for tier ranges, and tier assignment. This entire problem will be solved in a single function based on the provided inputs.

Note: Each column of input X corresponds to a single player. The first row represents the score of the player, the second row corresponds to whether or not a bonus is assigned, and the third row represents the penalty assigned.

Score Adjustment: Let us discuss what needs to be done to adjust the scores of each player. The scores are adjusted per the following rules:

- If players earned a bonus (value of 1), their scores are increased by 5.
- If their penalty is 1, their scores are decreased by 5.
- If their penalty is 2, they are removed from the player pool.

When solving this portion of the problem, use the following strategy:

1. Using a `while` loop:
   (a) Apply the appropriate changes to each score.
   (b) Count the number of players removed from the pool. Name this variable `removed`.
   (c) Remove the appropriate players from the pool.

[1] This problem is courtesy of Marko Jaćović.

2. Use `fprintf` to display the number of players that remain in the pool in your command window. Sample output:

```
>> Out of the original 50 players, 37 remain.
```

Determining Metrics for Tier Ranges: The mean and standard deviation of the adjusted scores will be calculated so that tiers may be assigned based on the pool of players that are remaining. Use the following coding strategy:

1. Using `for` loops:
   (a) Calculate the Mean of the scores. Name this variable `Xmean`.
   (b) Calculate the Standard Deviation of the scores. Name this variable `Xstd`.
   Note: Do not use `mean`, `sum`, `std`, *or* `var`.
2. Use `fprintf` to display the mean and standard deviation values. Sample output:

```
>> The mean of the scores is 50.270, standard deviation is 29.467.
```

Tier Assignment: Finally, tiers will be assigned to each player that remains based on the following rule:

$$Tier(x) = \begin{cases} 1, & \mu + \frac{4}{3}\sigma \leq x \\ 2, & \mu + \frac{1}{3}\sigma \leq x < \mu + \frac{4}{3}\sigma \\ 3, & \mu - \frac{1}{3}\sigma \leq x < \mu + \frac{1}{3}\sigma \\ 4, & \mu - \frac{4}{3}\sigma \leq x < \mu - \frac{1}{3}\sigma \\ 5, & x < \mu - \frac{4}{3}\sigma \end{cases}$$

where x is the score, μ is the mean, and σ is the standard deviation. Use the following coding strategy:

1. Create an output variable `Y` in which the first row is the score, and the second row is the resulting tier assigned. The columns of `Y` correspond to each player considered.

   Note: This refers to preallocating the output matrix.
2. Using a `for` loop, assign the tier values to each player.
3. Using a `for` loop and `fprintf`, list the number of players in each tier in the command window. Sample output:

```
>> The number of players in each tier is as follows:
Tier 1: 1 player
Tier 2: 8 players
Tier 3: 10 players
Tier 4: 10 players
Tier 5: 8 players
```

   Note: If there is only 1 player, the print statement should reflect that properly. If there are 0 players, then use the plural form , that is players.

Function Structure: No additional user defined functions may be used in solving this problem. All usage of `fprintf` will write to a text file named `output.txt`.[2] The file identifier is to be named `fid`. The function will also provide an output string variable named `textfile` which contains the entire contents of the text file. The text file must match the sample output provided in the previous sections.

The function used in solving this problem is required to take the following form:

```
function [removed,Xmean,Xstd,Y,textfile] =
videogame_rankings(X)

% Open Read/Write capable Text file
fid = fopen('output.txt','w+');
% Do not modify above this line.



% Enter solution here.


% Do not modify below this line.
fclose(fid);                          % Close text file
textfile = fileread('output.txt'); % Save file as a string

end
```

The contents of the `output.txt` file must be formatted as follows:

```
Out of the original 50 players, 37 remain.
The mean of the scores is 50.270, standard deviation is 29.467.
The number of players in each tier is as follows:
Tier 1: 1 player
Tier 2: 8 players
Tier 3: 10 players
Tier 4: 10 players
Tier 5: 8 players
```

Note: The formatting must match the above sample exactly including the capitalization, the punctuation, and the spacing. The singular use of 'player'needs to be used where appropriate. The actual values of course will vary based on the generated input vector X.

[2] Use the MATLAB help function to learn how to print to a file rather than to the command window.

Submission Instructions

- Submit the functions as separate .m files via BBLearn under the link called "Programming assignments" by December 2, 2015, 11:59 pm. Note that you may upload your work more than once; the latest version will be evaluated.
- In addition, you must submit and test all your answers on Cody by December 2, 2015, 11:59 pm. The submission site will be up and running by Saturday afternoon at the latest. we will send out an email to the class once the site is up.