CS 1132 Spring 2017 Assignment 2 due 4/19 at 11:59 pm

Adhere to the Code of Academic Integrity. You may discuss background issues and general strategies with others and seek help from course staff, but the implementations that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is never OK for you to see or hear another student's code and it is never OK to copy code from published/Internet sources. If you feel that you cannot complete the assignment on you own, seek help from the course staff.

When submitting your assignment, follow the instructions summarized in Section 3 of this document.

Do not use the `break` or `continue` statement in any homework or test in CS1132.

1 Data Exploration: Tennis Matches in 2014

This assignment will work through real data from a list of all tennis matches in the Association of Tennis Professionals (ATP) Tour for the year of 2014. The given data file has been pre-processed only minimally, so you will get to work with a set of data that is not perfectly simplified—a realistic scenario! You need to process the file—import data, manipulate, and compute—and then answer a set questions and produce a couple visualizations. These answers and visualizations must be achieved *programmatically* and not hard-coded. I.e., you need to write code to read the data and solve the problems and not pre-determine the answers by personally reading the data file and then hardcoding the answers and graphs. In other words, if we run your functions using a different set of data that follows the same file format (e.g., data from a different year), your functions should provide correct answers based on the actual data set used.

Completing this assignment will help you reinforce your knowledge about manipulating cell arrays and strings and will further give you the opportunity to explore some of MATLAB's plotting tools. You will also practice using MATLAB documentation to look up functions and their use.

1.1 Data file

The data file named `tennis_matches.txt` will be used for this assignment. This text file contains 17 columns representing certain aspects of a tennis match which we will explain below. Each row represents a tennis match between two players. Don't be intimidated by the amount of information contained in this data set! We will not be using all information stored in `tennis_matches.txt`; we have left some of them for you to explore in case you're interested.

Here is a brief description of each column contained in the data set:

- Tournament name - This refers to the tournament where the tennis match was played. The tournament name is usually the name of the city where the tournament was held.
- Surface - This refers to the surface of the tennis court, which can only be one of Hard, Clay, or Grass.
- Winner Seed - Tennis tournaments are seldom played in round robin format. Instead, a bracket system is used wherein tennis players are paired up and only the winners of their respective matches advance to the next round where they will be paired up again with another player. To ensure that top players only play against other top players in the latter—not early—stages of the tournament, top players are assigned seeded positions in the bracket and seeded players do not play against one another in the early rounds. This column refers to the seeded position of the winner of the tennis match. What you should know is that this column can contain positive integers only and has many blanks because not all players are seeded at a tournament.
- Winner Name - A string containing the name of the winning player.
- Winner Hand - A character, which could either be 'L' or 'R' representing the dominant hand that the player uses to hold the racket.

- Winner Height - A number representing the player's height. Assume that this is constant throughout the year. In other words, if the player's height at a certain match was recorded as unknown, we will assume that in the other matches that the player has played, the height is also unrecorded.
- Winner Country - A string representing the three letter code of the winning player's country.
- Winner Age - A real number (non-integer) representing the age of the winning player. This varies throughout the year.
- Loser Seed, Loser Name, Loser Hand, Loser Height, Loser Country, Loser Age - Same information as above but for the losing player.
- Score - This is a string representing the score of the tennis match.
- Round - This is a string representing the stage of the tournament at which the match was played. For example, 'F' signifies that it's the final round match of the tournament while 'SF' signifies that it is one of the 2 semi-final matches of the tournament, etc.
- Duration - This is a number representing the length of the tennis match.

Take a look at the data file using any text editor, e.g., the MATLAB editor (double-click on the filename within the MATLAB desktop).

If data from another year will be provided, you can expect that they will have the same format as described above.

1.2 Required, allowed, and forbidden functions

You will submit five function files: `exactMatch.m`, `readTennisData.m`, `extractPlayerData_countryStats.m`, `summarize_playerMatchStats.m`, `computeHead2Head.m` and the script `assignment2.m`. The given file `cellstr2str.m` is a completed function that you can download and use.

Use only the built-in functions listed below for handling characters, strings, and files; you may not need all of them. You can of course still use general built-in functions not related specifically to strings and files, such as `length`, `zeros`, `cell`, etc. You can use:

- `fopen`, `fclose`, `feof`, `fgetl` for file handling,
- `textscan` (explained below) for parsing (separating into parts) a string, and `strcmpi` for comparing strings.

You must *not* use built-in functions `find`, `strfind`, `findstr` in this assignment.

1.2.1 Matching strings

Implement the following function:

```
function idx= exactMatch(C, s)
% C is a 1-d cell array of strings with unique entries.
% Find the cells in C that match
%   string s exactly other than case.
% idx is a vector of the indices of C where the matches occur.
% E.g., if  C is {'Matt'; 'uses'; 'Matlab'; 'on'; 'a'; 'mAt'}
%       and  s is 'Mat', then idx is [6].
% If there is no match then idx is [].
% DO NOT use built-in functions strfind, findstr, or find.
```

Your function implementation should be efficient. In other words, once the match is found, your code should not continue on to check the remaining elements of the cell array. Built-in function `strcmp` is case-sensitive while `strcmpi` is case-insensitive. In this assignment we will use case-insensitive comparisons.

For this and all other functions that you write where a function specification—function header comment—is given, be sure to copy *exactly both* the function header *and* the function comment into your function file. I.e., the function specification should be complete in your file.

Be sure to work on and test your functions one at a time! The example in each function comment is a test that you can use; you should create other tests as well. Make effective use of this function in the remaining parts of the assignment.

1.2.2 Reading the data set

We will now write a function to transfer the data that is stored in `tennis_matches.txt` into a MATLAB array to enable us to manipulate the data later on. Since the information contained in `tennis_matches.txt` is a mixture of strings and numbers, a cell array would be the most ideal array type to store the information. Implement the following function as specified:

```
function tennis_data = readTennisData(filename)
% This function reads the contents of filename into cell array tennis_data.
% filename is a text file in which the first line stores the headers of the
%   dataset while the remaining rows contain the data to be stored in
%   tennis_data.  Each column of data in filename is delimited by ',' and
%   it is possible that there is no data between delimiters.
% tennis_data is a 2D cell array where each row corresponds to a row in
%   filename excluding the the header and where each column corresponds to
%   a column of data in filename.  If the data is missing for a particular
%   entry, replace it with -Inf.
```

Before implementing this function, inspect the data set first to see how it looks. As you can observe, the first line of the data set contains the header names of the columns but we are only storing the information contained from line 2 onwards. Therefore, your function should read the file line by line (but discard/ignore the first line read). Each line of this data set contains data items delimited by ','. You will convert each line of the data set into a 1-d cell array where each cell stores one data item and any missing data is given the value `-Inf`. `Inf` is MATLAB's built-in signal value for infinity.

The built-in function `textscan` is useful for parsing strings that represent both numeric and text information and that use a known delimiter. Here is an example for parsing a string delimited by commas:

```
str= 'abc,5,-3,h,,defg';  % note that the 5th data item is missing
data= textscan(str, '%s %f %f %s %f %s', 'delimiter', ',', 'EmptyValue', -Inf);
```

In the call to `textscan` above ...

- The first argument is the string (variable) to parse, `str`.
- The second argument indicates the expected format of the individual data items in the string: `'%s %f %f %s %f %s'` indicates *string number number string number string*, just like the format sequences used in `fprintf` statements.
- the third and fourth arguments above say that the delimiter is a comma.
- The fifth and sixth arguments above say that any missing value is to be replaced by `-Inf`.

`textscan` returns a 1-d cell array of the data items, but it deals with strings in a peculiar way.[1] Continuing with the above example, `data` would store the *nested* cell array `{ {'abc'}, 5, -3, {'h'}, -Inf, {'defg'} }`, i.e. each string data item is stored as a cell array within a cell array. To circumvent this problem, we have provided the function `cellstr2str` which removes the nesting. Using `cellstr2str` on `data` above would return the non-nested 1-d cell array `{ 'abc', 5, -3, 'h', -Inf, 'defg' }`. Read the function header of `cellstr2str` to learn how to use it.

Finally, how do you build a 2-d cell array given two 1-d cell arrays? Use *concatenation*! Here's an example:

[1] `textscan` is a useful function for handling delimited data but has many peculiarities. For future use (outside of CS1132) you would want to read through MATLAB's documentation to learn additional options and do some experiments!

```
data1= { 'abc', 5, -3, 'h', -Inf, 'defg' };
data2= { 'aaa', 5,  6, 'h',    2, 'xyzzz' };
arr= [data1; data2];
```

Note the use of *square brackets*, not braces, for the concatenation operation. If you use braces, then you would end up with a nested cell array instead of the requested *2-d* cell array.

Make sure that this function works correctly before proceeding with the other functions. The output of this function will be used as an input for the remaining functions you have to write and it would be important for you to be able to test those functions using the correct output from `readTennisData`.

1.3 Extracting Player Data and Country Statistics

As you can observe from our data set, we have a list of all tennis matches during the calendar year but we do not have a list of all the players who competed! We therefore want to write a function that compiles a list of all players and summarizes what we know about each player. In addition, we are also interested in keeping track of the countries represented by these players. Implement the following function as specified:

```
function [playerData,countryStats] = extractPlayerData_countryStats(tennis_data)
% This function extracts player data and country statistics from
% tennis_data which is a 2D cell array that represents the output from
% readTennisData.

% playerData is a 2D cell array where each row corresponds to data for a
% particular player. The first column represents the player's name while
% the remaining columns represent the player's country, dominant hand,
% age, and height. The player age recorded in the original data set refers
% to the age of the player at a particular tournament. Since this is not
% constant across all tournaments, we will round the player age to the
% nearest integer when stored in playerData.

% countryStats is a 2D cell array where each row corresponds to data for a
% particular country. The first column represents the country name while
% the second column represents the number of tennis players from that
% country.
```

For efficient implementation, make use of the function `exactMatch` that you implemented. There should be no duplicate entries for any particular player!

For this function you may find that you want to access a whole row or a whole column of a 2-d cell array. The syntax for accessing a row (or a column) is same whether you are working with a 2-d simple array or a 2-d cell array. For example the expression `M(4,:)` accesses the fourth row of the 2-d array `M` whether `M` is a simple array or a cell array.

1.4 Summarizing Player Match Statistics

Now that we have a list of all players who competed during the calendar year, we summarize how well each player performed by determining how many matches they won and lost. As the surface of the tennis court makes a difference, we also want to obtain a breakdown of this statistic according to the court surface. Finally, we can also tabulate how many tournaments each player won during the year. Note that there are more players than tournaments and that many top players won multiple tournaments during the year, which means that many players did not win a single tennis tournament during the calendar year. In order to determine whether a player won the tournament, check the entry under the column for 'Round' for the string 'F', i.e., the final round of the tournament (not the earlier rounds such as semi-final , quarter-final, etc.).

Implement the following function as specified:

```
function playerMatchStats = summarize_playerMatchStats(playerData,tennisData)
% This function summarizes the number of matches won and lost by each
% player for each of the three surfaces. playerData is a 2D cell array
% where each row represents the data for each player. The player's name is
% contained in the first column. This is precisely the output from
% extractPlayerData_countryStats. tennisData is a 2D cell array
% representing the original data set and is the output of readTennisData

% playerMatchStats is a 2D array where each row corresponds to the player
% represented in the corresponding row in playerData. The columns of
% playerMatchStats represent the following:

% column 1 - total number of matches won
% column 2 - total number of matches lost
% column 3 - total number of matches won on Hard court
% column 4 - total number of matches lost on Hard court
% column 5 - total number of matches won on Clay court
% column 6 - total number of matches lost on Clay court
% column 7 - total number of matches won on Grass court
% column 8 - total number of matches lost on Grass court
% column 9 - total number of tournaments won

% The total number of tournaments won can be obtained by checking if the
% player won the "Final" stage of a tournament as indicated by an 'F' in
% column 16 of tennisData.

% Make effective use of function exactMatch.
```

As you are thinking about how to implement this function, bear in mind that you should aim to write code that is as efficient as possible. This is because in real scenarios, the data set at hand could have a much larger number of rows and columns than the one we're dealing with in this assignment. Writing efficient code means that you don't have to wait for a long time for your data set to be processed! After all, there is only so much time in this world.

A straightforward way of implementing this function would be to go over each line in the data set for each player in the list we compiled above. If there are 3000 lines in the data set, i.e., 3000 tennis matches, and if there are 300 players who competed on tour, then that would mean 900000 iterations in your code! Try to see if you can be more efficient than that—do you really need all those iterations?

1.5 Computing the Head-to-Head Record Between Two Players

When tennis analysts preview an upcoming match between two tennis players, a statistic that they often consider is the "Head-to-Head" record between the two players. This statistic indicates how many times player A has beaten player B and vice versa. Because the surface of the tennis court also has an impact on how the tennis match will turn out, analysts are also interested in how the players fared against each other depending on the surface. Implement the following function as specified:

```
function H2H = computeHead2Head(tennisData,player1,player2)
% This function summarizes the Head to Head record between two tennis
% players. tennisData is a 2D cell array which is the output from
% readTennisData. player1 and player2 are strings representing the names of
% the players.

% H2H is a 2D array where column 1 refers to the number of matches player1
% has won against player2 while column 2 refers to the number of matches
% player2 has won against player1. H2H has 4 rows as follows:

% row 1 - matches played on Hard court
% row 2 - matches played on Clay court
% row 3 - matches played on Grass court
% row 4 - all matches played

% This function also neatly prints this table to the command window

% Note that it is possible for 2 players to not have played against each
% other during the calendar year.
```

In the event that two tennis players have not played against each other during the calendar year, all the entries in matrix `H2H` will be zero. In order to test your code on a non-trivial case, you can check the Head-to-Head record between prominent players such as Novak Djokovic, Rafael Nadal, and Roger Federer.

1.6 Answer questions on and make visualizations from the data

Add code to `assignment2.m` to answer the following questions and make the requested visualizations (draw figures). Your code should print a descriptive answer to the Command Window so that it is clear what question is being answered. Include the question number in the screen output. Observe that question 1 has been answered for you in the provided code as an example.

You should not need to write additional functions to answer the questions below. Make effective use of the functions we have developed thus far.

1. How many players are there on tour?
2. How many countries are represented on tour? Also draw a *horizontal* bar graph to show the number of players from each represented country. The y-axis should show the country names.

   Here's a *vertical* bar graph example:

   ```
   xdat= 1:4;  ydat=[7 11 9 5];
   bar(xdat, ydat)  % 4 bars with heights specified by ydat
                    % The bars are located at x coordinates xdat.
   ```

   The bar graph in MATLAB uses numeric axes. If you want to have text labels (country names) on the horizontal axis, then you need to add the following code:

   ```
   barNames= {'Name1' 'Name2' 'Name3' 'Name4'};  % cell array containing the bar names
   set(gca, 'XTick', xdat, 'XTickLabel', barNames);
            % set and gca are built-in functions
            % XTick and XTickLabel are built-in property names of the graph
   ```

   Producing a horizontal bar graph requires code that is very similar to that above. You simply use `barh` instead of `bar` and set the properties of the y-axis instead of the x-axis for the country names. You can read MATLAB's documentation to learn more.

   Be sure to label the x- and y-axes and give your graph a title. Note that the country names may overlap when viewed in the default size figure window. Do not worry about this—one can always maximize the figure window when viewing the graph. *Optional:* you can solve the overlap problem by including

the `'FontSize'` property in the `set` statement above and giving it a smaller value than 10, which is the default font size. Again, search MATLAB's documentation to learn more if you are interested.

3. How many tournaments did Rafael Nadal win during this calendar year?
4. What is the head to head record between Novak Djokovic and Rafael Nadal?
5. How many matches are played on each type of surface? Draw a pie chart to visualize this. Use MATLAB's documentation to learn how to draw a pie chart. Begin your graphics code with the command `figure` to start a new figure window (so that the previous bar graph is not replaced). Be sure to label the graph appropriately.
6. How many times did a top seed at the tournament win the tournament?
7. What fraction of players are left-handed?
8. How many tournaments had champions who are left-handed?

2 Self-check list

The following is a list of the minimum *necessary* criteria that your assignment must meet in order to be considered *satisfactory*. Failure to satisfy any of these conditions will result in an immediate request to resubmit your assignment. Save yourself and the graders time and effort by going over it before submitting your assignment for the first time.

Note that, although all of these are necessary, meeting all of them might still not be *sufficient* to consider your submission satisfactory. We cannot list everything that could be possibly wrong with any particular assignment!

- Δ Comment your code! If any of your functions is not properly commented, regarding function purpose and input/output arguments, you will be asked to resubmit.
- Δ Suppress all unnecessary output by placing semicolons (;) appropriately. At the same time, make sure that all output that your program intentionally produces is formatted in a user-friendly way.
- Δ Make sure your functions' names are *exactly* the ones we have specified, *including* case.
- Δ Check that the number and order of input and output arguments for each of the functions matches exactly the specifications we have given.
- Δ Test each one of your functions independently, whenever possible, or write short scripts to test them.
- Δ Check that your scripts do not crash (i.e., end unexpectedly with an error message) or run into infinite loops. Check this by running each script several times in a row. Before each test run, you should type the commands `clear all; close all;` to delete all variables in the workspace and close all figure windows.

3 Submission instructions

1. Upload files `exactMatch.m`, `readTennisData.m`, `extractPlayerData_countryStats.m`, `summarize_playerMatchStats.m`, `computeHead2Head.m` and `assignment2.m` to CMS in the submission area corresponding to Assignment 2 before the deadline.
2. When the scores are released, read the grader's feedback carefully.
3. If you need to resubmit, fix all the problems and go back to Step 1! Otherwise you are done with this assignment. Well done!