Drill Problem #1

Function Name: blackOut

Inputs:

1. (*char*) A string identifying the name of a text file
2. (*cell*) A list of words to be removed from the document

Outputs:

(*none*)

File Outputs:

1. A text file with asterisks in place of the blacklisted words occurring in the text file

Function Description:

Congratulations! You have just been hired by a new, super-secret organization for your superior MATLAB skills. This organization needs you to go in and censor their documents. The company will give you the name of a file and a list of words to be censored. Find these words in the file, ignoring the case of the words as they appear in the file, and replace them with a string of asterisks that are the same length as the word, maintaining the location of spaces in the altered string.

For example, if `'Jen Alpha'` were in the cell array list of words to be removed, `'Jen Alpha'` would become `'*** *****'`. This means the statement `'Jen alpha breaks into the safe'` would be written in the new text file as `'*** ***** breaks into the safe'`.

The name of your output file should be the name of your input file with `'_CENSORED'` added onto the end. So `'text.txt'` would become `'text_CENSORED.txt'` for the output file name.

Hints:

– A helper function may be useful to locate the position of the word to be censored.

Drill Problem #2

Function Name: accounting

Inputs:

1. (char) Spreadsheet filename

Outputs:

(*none*)

File Outputs:

1. An excel file with the corrected spreadsheet

Function Description:

You've been hired by the Mob. Yes, that Mob.

Your job is to take some books they've cooked and turn them into more readable, understandable spreadsheets so when the Feds show up and demand it all as evidence it'll be nice and pretty, and they can worry their little heads off about where the couple of dimes here and there went; all the while Johnny takes the duffels of laundered money out the back and stashes them- AHEM. Right. Yeah. So. Books.

Take an "encoded" (see also: gibberish) book that the Mob gives you and turn it into a more readable format. They've decrypted the headers for the "Item", "Quantity", and "Per-unit Price" columns so you can do the math they require of you. The columns have been left encrypted for how much of it was clean, how much went to a given day's tab, how much was "blood money"- AHEM.

Create a new spreadsheet with the following column headers: "Item", "Quantity", "Per-unit Price", "Per-unit Tax", and "Total Price" (which have been decoded for you). Underneath each column, either place the already existing columns or calculate the respective values for each item. Assume a tax of 8%, so the "Per-unit Tax" column will be "Per-unit Tax" = 0.08 * "Per-unit Price" and "Total Price" = ("Per-unit Tax" + "Per-unit Price") * "Quantity". Round the values in the "Per-unit Tax" and "Per-unit Price" columns to the hundredths decimal place (so, round to the nearest cent).

The output file should be named the same as the input file but with '_fixed' before the file extension. E.g. 'example.xls' would become 'example_fixed.xls'.

Notes:

- It is very highly recommended that you DO NOT use iteration to solve this problem.
- Keep the items in the order they came in, just move the column (if it needs to be moved).
- Each item will be unique.
- The mob has provided a sample "back to school shopping list" spreadsheet that will help you visualize what it is your code will do.

Drill Problem #3

Function Name: healthyHabits

Inputs:

1. (*char*) string representing a recipe
2. (*char*) filename of a .txt file containing a cookbook
3. (*char*) filename of an excel file containing ingredients and their nutrition facts

Outputs:

1. (*cell*) 5x1 cell array containing nutrition facts of input recipe

Function Description:

You've decided you want to eat healthier—well, you at least want to keep track of what you eat! Write a function that takes in a recipe name, a cookbook .txt file containing various recipes, and an excel file containing ingredients at hand and their nutrition facts, and outputs the nutrition facts of the input recipe in a 5x1 cell array. To calculate each nutrition fact, multiply the quantity of the ingredient with its respective nutritional information and sum the total for each ingredient. For example, pulling the recipe from the text file and pulling the ingredients from the excel file for the recipe 'Egg In A Nest':

```
'cookbook.txt'
Egg In A Nest
- 2 eggs
- 1 slice bread
```

'pantry.xlsx'

Food	Calories	Total Fat	Carbs	Protein
eggs	71	5	0	6
bread	70	1	13	3

The total calories will be (2*71) + (1*70) = 212 (then perform similar calculations for total fat, carbohydrate, etc.). Your output should be a 5x1 cell array of the recipe's nutrition facts.

Therefore, the function call:

```
out1 = healthyHabits('Egg In A Nest','cookbook.txt','pantry.xlsx')
```

will result in the output:

```
out1 => {'Egg In A Nest';
         'Calories: 212';
         'Total Fat: 11 g';
         'Carbs: 13 g';
         'Protein: 15 g'}
```

Notes:

- When calculating nutrition facts, simply pull out each number from each ingredient line. Don't worry about ounces, cups, slices, etc.

- Each ingredient in the pantry will have a unique name such that no ingredient name will be a substring contained in another ingredient name. Further, each recipe will only appear in the cookbook once.
- Each output cell array will always be in the format of:

```
{<specified recipe>;
'Calories: <num_calories>';
'Total Fat: <grams_fat> g';
'Carbs: <grams_carbs> g';
'Protein: <grams_protein> g'}
```

  Where the last four entries will match with the column headings in the excel file.
- The excel file is not guaranteed to have the columns in any particular order.
- Your code should be case insensitive, meaning it should work no matter the case of the recipe, pantry, or cookbook.
- There will always be a space between recipes in the cookbook.
- There may be a double-digit amount of any single ingredient called for in a recipe.
- As always, outputs must match exactly with the solution file outputs or you will get zero credit.

Hints:

- You may find the `strfind()` function useful.

Drill Problem #4

Function Name: itsaMe

Inputs:

1. (*char*) String containing the name of the grand prix Excel Spreadsheet

Outputs:

1. (*char*) Congratulatory message for winner

File Outputs:

1. A new Excel file (<original_filename>_Results<original_fileExt>) with an additional "Finish" column containing the racers in the order that they finished

Function Description:

Let's-a play Mario Kart! Well, let's-a write a code to tell us who-a won-a Mario Kart! Given an Excel file containing data from each race of a grand prix (series of races), rank the players in order of most points earned and write this list into a new column of the file. Additionally, output a congratulatory message containing the name and rank of each player. Here's an example:

The file `'mushroomCup.xls'` contains the following information:

Place	Points	Race 1 of 4	Race 2 of 4	Race 3 of 4	Final Race
1st	10	Mario	DK	DK	Bowser
2nd	7	Yoshi	Yoshi	Toad	Peach
3rd	4	Peach	Toad	Mario	Luigi
4th	3	Bowser	Bowser	Bowser	DK
5th	2	DK	Peach	Luigi	Yoshi
6th	1	Luigi	Luigi	Yoshi	Mario
7th	0	Toad	Mario	Peach	Toad

In this prix, Mario earned 10 points in the first race, 0 in the second, 4 in the third, and 1 in the final race, for a total of 15 points. Yoshi earned 7 points in the first and second races, 1 point in the third, and 2 points in the last race for a score of 17. The rest of the racers totals are shown below:

Mario	15
Yoshi	17
Peach	13
Bowser	19
DK	25
Luigi	8
Toad	11

DK earned the most points, so the output string should read:

```
'Congratulations! DK wins with a total of 25 points!'
```

And the updated file `'mushroomCup_Results.xls'` should contain:

Place	Points	Race 1 of 4	Race 2 of 4	Race 3 of 4	Final Race	Finish
1st	10	Mario	DK	DK	Bowser	DK
2nd	7	Yoshi	Yoshi	Toad	Peach	Bowser
3rd	4	Peach	Toad	Mario	Luigi	Yoshi
4th	3	Bowser	Bowser	Bowser	DK	Mario
5th	2	DK	Peach	Luigi	Yoshi	Peach
6th	1	Luigi	Luigi	Yoshi	Mario	Toad
7th	0	Toad	Mario	Peach	Toad	Luigi

Notes:

- The first column of the file will contain the places (as strings), and the second column will contain the point values (as doubles) that racers finishing in those places earn, but there can be any number of racers and races per file.
- Each race column will always contain every racer for the prix in the place rankings.
- The point values awarded for each place do not necessarily vary linearly (ex: 1st place gets 15 points, 2nd place gets 12 points, 3rd place gets 10 points, 4th and 5th get 5 points), and may vary per prix.
- The titles of each column are arbitrary, but they are guaranteed not to be doubles.
- You are guaranteed to not have a tie in the final totals.

Hints:

- The column titles and place column (first row and first column) do not need to be used in this problem.
- You may find both outputs of the `sort()` function useful.

Drill Problem #5

Function Name: webTracker

Inputs:

1. (*char*) String containing the name of the Excel Spreadsheet
2. (*char*) String that is the month of interest

Outputs:

1. (*char*) The IP address that spent the most time on the website in the month

Function Description:

After pitching your MATLAB skills to some recruiters at the internship fair, they were impressed and offered you a position with a company (it may or may not have been a specific government agency…we are not at liberty to disclose that information) that tracks web traffic data.

Write a MATLAB function called webTracker that will input the filename of a spreadsheet and the particular month of interest. The spreadsheet will contain three columns and an unspecified number of rows, the first column being the `'Date'`, second column being a user's `'IP Address'` (the unique signature a user's computer leaves when they visit a website), and the third column being the `'Minutes Active'` (the number of minutes that user spent on the website on that day during a single connection). The date will be in the format of either M/D/YYYY or MM/DD/YYYY format, with single or double digits depending on the month and day of interest. Then, determine which individual user spent the most time connected to the website during that month and output that user's IP Address in class char.

Example Spreadsheet:

Date	IP Address	Minutes Active
4/1/2014	20.46.212.230	713
4/1/2014	20.46.212.230	622
4/1/2014	143.25.213.52	856
4/2/2014	211.100.81.189	261
4/4/2014	145.38.165.199	440
4/5/2014	106.235.140.234	519
4/6/2014	106.235.140.234	670
4/6/2014	240.206.113.236	923

Notes:

- There may be multiple connections made by the same user on the same date.
- Input month format is January, February, March, April, etc (full English month name).
- You will definitely have data in the spreadsheet for the input month.
- There will not be a tie.
- You may *NOT* use the built-in function `unique()` or you will receive no credit.