You may notice that the O (output) in High Level File I/O is left out of this homework. That is because `xlswrite()` does not work on Macs so for convenience we do not require you to use this function, except on the ABCs. However, you will still need to know the semantics of this function on the tests. We have set up some of the functions such that the output is a cell array that could readily be written to an Excel file, we just don't require that you actually write any files.

Some Mac users have reported difficulty opening `sheet1.xlsx` and `sheet3.xlsx` for `sortBy()`. We are not sure why this is happening, but opening the file in Excel and resaving seems to fix the problem.

Happy Coding,
~Homework Team

Function Name: `cellSum`

Inputs:

1. *(cell)* A 1xN cell array

Outputs:

1. *(double)* The sum of all doubles in the cell array

Function Description:

Unfortunately, the sum function does not work for cell arrays in MATLAB. However, you can remedy that problem by writing your own function! The input cell array can be of any size and each element in the cell array can contain any data type we have learned about so far, including another cell. The only thing you are guaranteed *not* to find within a cell is another multi-dimensional cell array (meaning, it will be 1x1). For example, this is a valid test case:

```
{{3}, [4, 2], {'string'}, {{[1, 1]}}}
```

but this is not:

```
{{2, 4}, 2}
```

In the first case, the output should be `11` because `3 + 4 + 2 + 1 + 1 = 11`.

Notes:

- If there are no numbers in the cell array, output `0`.
- Only sum values of type double.

Function Name: `pawnShop`

Inputs:

1. *(char)* The shop inventory filename
2. *(char)* The shop log filename

Outputs:

1. *(cell)* Nx2 cell array containing your updated inventory
2. *(double)* Your net cash flow

Function Description:

You've always dreamed of running a pawn shop with your family. Now your dreams have become a reality (TV show)! The media attention your shop has gathered is now attracting hordes of famous people who want to come browse your shelves full of mystery and grandeur. Due to this influx of activity, you have decided to use MATLAB to keep track of your shop's activities.

Given a set of transactions ordered chronologically, update your inventory and calculate your net cash flow over the course of the transactions. The function's first input indicates the name of the Excel spreadsheet containing your shop's inventory and the second input indicates the name of the Excel spreadsheet containing the transactions. The inventory will always have the items in the first column and price in the second. In the second spreadsheet, the columns will always contain the customer's name, item he or she is interested in, whether he or she is buying or selling (`'Buy'` or `'Sell'`), and the price he or she is willing to pay or are offering, in that order. There will always be column headers in both files. Each transaction has its own row.

If the customer is buying an item that is not in the inventory, skip over that transaction. If the customer is buying an item that is in the inventory, compare the asking price with the inventory price. If the asking price is greater than or equal to the inventory price, sell the item at the asking price and remove it from your inventory. If the asking price is lower than the selling price, do not sell them the item.

If the customer is selling an item, you will offer half of what he or she initially asked for. The customer will always (reluctantly) part with the item. You will then add the item and the **customer's initial price** to the end of the inventory (because you want to make a profit).

The first output is the new inventory after processing all of the transactions in the log. This should be a cell array in the same format as the initial inventory (excluding headers) after it has been read in from the Excel file. New items should be appended to the end in the order of appearance in the log. Your net cash flow (second output) is the sum of your revenue from selling minus your expenses from buying.

Notes:

- An item will only appear in the inventory once. You will not be sold an item you already have in the inventory.
- Round all calculated prices to the nearest cent.

Function Name: `sortBy`

Inputs:

1. *(char)* An Excel filename
2. *(char)* A header to look for

Outputs:

1. *(cell)* The modified cell array

Banned Functions:

`sortrows()`

Function Description:

Imagine you have a set of Excel files that are organized either horizontally or vertically. That is, the headers of the data may either go down the first column or across the first row. Regardless of the layout, you want to be able to sort the data based on a given header. You will be given the name of an Excel file and a header to sort the data according to. This means that you must sort that column or row numerically or alphabetically, and re-index all the columns/rows accordingly. If the row or column you must sort by contains all doubles, then sort from smallest to largest. Otherwise, if it contains all chars, then sort alphabetically (from A to Z). If the headers are in the first row, you will be reordering the rows of the Excel file and if the headers are in the first column, you will be reordering the columns of the Excel sheet.

Notes:

- The column or row you must sort will not have mixed data types.
- The header string will only appear once.
- If the header is located in the upper left cell (row = 1, col = 1), then assume the first row of the spreadsheet is the headers.
- You are guaranteed to be given a valid header.

Function Name: `partyPlanner`

Inputs:

1. *(char)* The filename of a survey text file with course options and guest preferences

Outputs:

1. *(char)* A string of the most popular item

Function Description:

You are throwing a party and you want to be sure that you make some food that as many people as possible will like. To do that, you gave everyone a survey and recorded the results in a text file. Of course, counting votes is too menial a task so you decide to implore your fearsome MATLAB prowess to count them for you. Each line of the file is a single vote for a dinner item and you need to determine which item is the most popular. The output will be a string of the most popular item. In the case of a tie, you should output the item that appears first in the list.

Notes:

- The items will not be in any particular order.
- There will not be any extraneous characters on any lines.
- There can be any number of unique items.
- Checking for the same item should be case sensitive.

Hints:

- There is a reason this problem is on the cell arrays homework.
- You only need to iterate through the votes file once.

Function Name: `carShopping`

Inputs:

1. *(char)* An Excel file name
2. *(char)* Your first priority when buying a car
3. *(char)* Your second priority when buying a car

Outputs:

1. *(cell)* A sorted 4xM cell array of the best three cars

Banned Functions:

`sortrows()`

Function Description:

With your great resume and MATLAB skills, you've gotten an offer for an internship! Of course, you'll now need a car to get to work, and thankfully your parents are willing to give you a small loan. To decide on a car, you decide to write a function that sorts a spreadsheet of car data and outputs a cell array of the best three cars. The second and third inputs are the factors you consider the most important when selecting a vehicle, for example: year, mileage, or safety rating. You consider the second input to be much more important than the third so you should sort by that first. So if year is the most important factor and horsepower is the second most important factor, you will first sort by year. Then, within each year you will sort cars based on horsepower. After doing this, you will output the top three options to the final cell array. The first row of this cell array will have the original Excel file column headers and the remaining three rows will have the best three cars based on the input criteria, sorted from best to worst.

Notes:

- The top row of the spreadsheet will always be headers but are **not** guaranteed to be in a specific order.
- The categories you will be sorting by will always be quantitative, and in each category a larger value indicates a better car.
- In the case of a tie in both criteria, sort by order of appearance (`sort()` will do this automatically).
- All data is from carfolio.com.