Notice

For this week's homework, some problems can only be tested after "loading" certain `.mat` files as instructed in the `hw05.m` file. You can load these files three ways before running test cases: right click the `.mat` file in your directory and select "Load" OR double click on the `.mat` file in your directory OR use the built-in `load()` function (see function documentation: https://www.mathworks.com/help/matlab/ref/load.html). If you are writing a test script, it would be wise to use the `load()` function appropriately. **DO NOT** use the `load()` function in your function code! To test your function(s), load the `.mat` file first and then run the function with the specified inputs just as you have been doing up until now. The Autograder will automatically load the appropriate `.mat` files for you before grading your homework. If `load()` is included in your function, it will error, and you will not receive credit.

Happy coding!
~Homework Team

Function Name: `rottenFruit`

Inputs:

1. (*double*) 1xN vector of age of each fruit in days

Outputs:

1. *(double*) Total number of edible fruit

Background:

Your New Year's resolution was to start eating more fruit, but now that the semester's picking up it's hard to keep track of which of the fruits that you bought you can currently eat! You begin to worry because no one wants to eat a rotten apple or an unripe banana- until you realize you can use MATLAB to help you figure out how many of fruits you have are currently edible.

Function Description:

Write a function that takes in a vector of days of how old various pieces of fruit are and outputs how many fruits are edible. If the fruit is older than seven days, it is rotten and must be thrown away. If the fruit is less than two days old, it isn't ripe enough to eat, thus making it inedible. In other words, the edible days should be between 2 and 7, inclusive.

Example:

```
rottenFruit([3 1 5 9 7]) => 3
```

Because 3, 5, and 7 are all between 2 and 7 inclusive.

Notes:

- The input will always be positive.

Hints:

- What happens when you sum logicals?

Function Name: `fruitLottery`

Inputs:

1. (*char*) vector representing the 6 numbers on your lottery ticket
2. (*char*) vector representing the 6 numbers on the winning ticket
3. (*double*) The initial investment required to open a fruit store

Outputs:

1. *(logical)* A logical representing whether you acquired the initial investment

Background:

You want to fulfill your lifetime dream of opening a fruit store, but don't have the cash to do it. Desperate to get rich quick, you decide to buy a lottery ticket to try your luck.

Function Description:

This function will compare your lottery ticket number (first input) to the winning ticket number (second input) and calculate your winnings. The ticket numbers are composed of 6 positive integers separated only by dashes.

Rules for determining winnings from lottery ticket numbers:

1. Compare the first five integers in both numbers. For every integer in your lottery ticket number that matches **any** of the first five integers in the winning lottery ticket number, add $100,000 to your total prize.
2. If the last number matches, then double the prize amount determined by the first 5 numbers.
3. Compare your earnings to the required initial investment, and output a true if your earnings are greater than or equal to the investment, and a false otherwise.

Example:

Bought lottery ticket	→	`'13-45-33-19-29-8'`
Winning lottery ticket	→	`'29-10-6-13-41-8'`
Maximum prize amount	→	`400000`

Notes:

- The first five numbers of the winning ticket can match the first five numbers of the bought ticket in **any** order.
- Each ticket will contain 6 **unique** integers (no repeated integers within a single ticket number).
- There will be no extraneous characters in the inputs.
- The output needs to be a logical to receive full credit.
- You can use str2num() on a space separated character vector of numbers to get a vector of doubles.

Hints:

- Adding a 1 to a logical will turn trues into 2 and falses into 1.

Function Name: `fruitDecryption`

Inputs:

1. (*char*) A vector of the encrypted message

Outputs:

1. *(char*) A vector containing the secret message

Background:

You have recently joined the **A**ssociation of **P**eople **P**romoting the **L**ove and **E**njoyment of Fruit**S** (APPLES), an underground society dedicated to the love and support of fruits worldwide. As this is a secret organization, any and all information regarding the organization's details are encrypted using a secret system. You quickly realize that deciphering these messages by hand is rather bothersome, and decide to write a MATLAB function that will help you uncover the secrets.

Function Description:

Write a function that will take an encrypted message and decipher it by following rules.

1. Remove any numbers, punctuation characters, and spaces.
2. Move all uppercase letters to the beginning of the vector, followed by all lowercase letters (while preserving the order of both).
3. Make any remaining characters lowercase.
4. Remove any character that comes immediately AFTER one of the character in the word "fruit" (`'f'`, `'r'`, `'u'`, `'i'`, and `'t'`). If the last letter is one of these characters, you can ignore it.

Example:

```
Input: 'in3Mt%.pef90wrYFI:1cZ)u AVn#iOR21 Adt0s'
> 'inMtpefwrYFIcZuAVniORAdts' (removes non alphabet characters)
> 'MYFIZAVORAintpefwrcunidts' (reorders values based on capitalization)
> 'myfizavoraintpefwrcunidts' (lowercase all values)
Output: 'myfavoritefruit'
```

Notes:

- Follow the order given above.
- Note above how `'i'` and `'z'` were both removed because each follows a letter of 'fruit'. The deletions can be consecutive.

Function Name: `fruitEncryption`

Inputs:

1. (*char*) MxN array of uncapitalized characters

Outputs:

1. *(char*) updated MxN array of characters

Background:

You finally decoded the secret messages for APPLES, and you are about to send a message back, when you discover that the evil fruit spies have cracked your code! You have an urgent message to send, and you have to quickly figure out a new way to encrypt this message. To throw the spy off, you decide to incorporate arrays.

Function Description:

Write a function that takes in a character array and encrypts it with the following steps:

1. Shift the characters in the even rows by the number of columns in the array.
2. Shift the characters in the odd columns by the number of rows in the array, but backwards (towards `'a'`).
3. Swap the top half of the array with the bottom half.

Note:

- Follow the order of shifting given above.
- The alphabet should 'wrap' around.
- When dividing the array in half, use `floor()` to account for the possibility that the array could have odd dimensions.

Hint:

- The logic behind shifting character values in this problem is exactly the same as the shifting logic from `coldWar()`.

Function Name: `fruitThief`

Inputs:

1. *(logical)* Vector of suspect #1's answers to a lie detector
2. *(logical)* Vector of suspect #2's answers to a lie detector
3. *(logical)* Vector of suspect #3's answers to a lie detector
4. *(logical)* Vector of suspect #4's answers to a lie detector

Outputs:

1. *(char)* Sentence stating which suspect stole the fruit

Banned Functions:

`isequal()`, `isequaln()`

Background:

Fruit has mysteriously started disappearing from your floor's kitchen. You and the other residents of your floor have decided that enough is enough and you begin your search for the fruit thief. Using your expert detective skills you cut your suspect pool down to four people and determine that the thief is working alone (only one of your suspects is the fruit thief). You give each suspect a lie detector test and use the results to find which of the four suspects is lying. Each suspect who is telling the truth will give a corresponding yes or no (`true` or `false`) answer to each question, while the suspect who is lying will not corroborate *at least one* of his/her answers with the other three. With your newly found MATLAB skills, you decide to write code to assist you in finding the thief.

Function Description:

Each input to the function is a logical vector corresponding to the answers a suspect gives on the lie detector. Each element of the vectors represent a different question. Three of the suspects will have the exact same answers, but one suspect's answers will be slightly—or completely—different than the others' answers. Using your knowledge of logical indexing and masking, determine which of the four suspects is lying, and thus, is the fruit thief.

The output string will be of the form `'Suspect #<num> is the fruit thief.'`, where `<num>` corresponds to the suspect number who stole the fruit. The number is determined by the input order.

Notes:

- The suspect who lied, and stole the fruit, will have *at least one* answer that is different from the other suspects' answers, but could differ up to every answer.
- You **may not** use the `isequal()` function to code this problem. Use of the `isequal()` function will result in zero credit for this problem. However, the use of `isequal()` to check that your output matches the solution outputs is encouraged.
- The output string ends with a period. Do not forget that period!

EXTRA CREDIT

Function Name: `fruitMarket`

Inputs:

1. (*char*) List of three fruits you want to buy, capitalized, comma separated (with no spaces)
2. (*char*) Character array of capital letters representing fruit market stalls

Outputs:

1. *(char*) Minimum cost of buying all your fruit

Background:

Modern day shoppers always struggle to find the cheapest prices in a market. With today's extensive buying options, shoppers like you need an optimal way to find the best deals.

Function Description:

Given a comma separated list of fruit that you want to buy, and a character array of the fruit that each stall sells in the market, calculate the **minimum** cost of buying all of the fruit on your list.

Each letter in the character array indicates which fruit the stall sells (based on the first letter of the fruit's name). The product (multiplication) of the character's coordinates determines the price of the fruit sold by that stall. The character array below is an example of the fruit sold at a fruit market.

```
'BKAO'
'OGPL'
'MTMB'
```

Banana is sold at two stalls: the stalls with coordinates `[1,1]` and `[3,4]`. Hence, you can buy bananas for `$1` or `$12` dollars each. You should output your final cost as a string, without any decimals: `'$<dollars>'`.

Example:

```
list = 'Banana,Apple,Orange'
market =    'BKAO'
            'OGPL'
            'MTMB'
>> cost = fruitMarket(list, market)
cost = '$6'
```

Notes:

- You may have repeat fruits in your list which means you are purchasing more than one of that fruit.
- If there are repeat fruits in the character array, find the minimum cost.

Hints:

- You can get two outputs from `find()`.