Function Name: `ripTacoBell`

Inputs:

1. *(double)* An NxM array of positive integers

Outputs:

1. *(double)* An NxM modified array of numbers

Background:

Why was six afraid of seven? Because seven eight nine! Why did seven eat nine? Because seven couldn't buy their #9 (Crunchwrap Supreme) at the Student Center Taco Bell! Rest in peace, Taco Bell.

Function Description:

Write a function that replaces all of the multiples of eight or nine in an array with seven, unless the multiple of eight or nine is also a multiple of seven, in which case it should be left alone.

Example:

```
>> inArray = [16 36 56;
               6  7 10]

>> outArray = ripTacoBell(inArray)

>> outArray => [7 7 56;
                6 7 10]
```

Function Name: `willage`

Inputs:

1. (*double*) A 9x9 array of numbers

Outputs:

1. *(double*) A completed 9x9 array of numbers

Background:

Your friend from the other school in Athens (u(sic)ga) frantically calls you while you are sitting in West Village having a peaceful meal. They tell you that they can't figure out the last number of their sudoku puzzle. They believe you can help them out, since you go to Georgia Tech. But alas, you have no time to figure the puzzle out by hand, so you set out to create a function with MATLAB to figure out the missing number.

Function Description:

Write a function that takes in a 9x9 array and find the zero, which represents the missing number in the puzzle. Then, replace the zero with the appropriate number to finish the puzzle.

Example:

```
    puzzle = [452391876         completedPuzzle => [452391876
              318675294                             318675294
              679428315                             679428315
              831564729                             831564729
              245907163                             245987163
              967213548                             967213548
              796852431                             796852431
              183749652                             183749652
              524136987]                            524136987]

>> completedPuzzle = willage(puzzle)
```

Notes:

- In sudoku, each row and each column has numbers 1-9 once, which means that each row and column should add up to 45.
- There will only be one missing number but it will not always be in the center row or column.

Hints:

- "`find()`" a function to help you look for the zero.

Function Name: `arRaysPizza`

Inputs:

1. *(char)* an MxN array representing pizza 1
2. *(char)* an MxN array representing pizza 2
3. *(char)* a single char representing a topping to look for

Outputs:

1. *(char)* an MxN modified pizza 1
2. *(char)* an (M-2)x(N-2) modified pizza 2

Background:

You and your friend are on campus and starving. Luckily, Ray's Pizza has opened a new knockoff location: arRay's Pizza, where they only make rectangular pies! Each of you decides to order a pizza with your favorite toppings. Upon receiving your food, you realize there was a mix up when ordering, and some of the topping from your pizzas have been switched! Luckily, you can use your MATLAB talents to fix the problem and enjoy a tasty meal.

Function Description:

Write a function that finds all the places where the input topping (third input) occurs in the character array representing the first pizza. Replace those locations in the first pizza with the corresponding toppings in the same locations in the second pizza. Then, add the input topping back into the second pizza at those locations. Finally, because your friend doesn't like crust, remove all the elements on the edges of the second array Return the two modified pizzas.

Example:

```
 pizza1 = ['abcd                     pizza2 = ['pppp
           Eafg                               pppp
           cbaA                               pppp
           Lmao]                              pppp']

>> [newPizza1, newPizza2] = arRaysPizza(pizza1, pizza2, 'a')


newPizza1 = ['pbcd                   newPizza2 = ['ap
             epfg                                 pa']
             cbpA
             lmpo']
```

Notes:

- The function should be case sensitive, so 'a' and 'A' are not the same
- Be sure to name the function correctly, including capitalization!

Function Name: `panera`

Inputs:
1. (*char*) A NxM character array representing the status of your order at Panera
2. (*char*) A 1xP vector representing your name

Outputs:
1. *(double)* The number of orders left until yours is ready

Background:

You are at West Village waiting for your delicious order of Panera mac and cheese. While you wait for your name to show up on the order screen, you realize that you could change the game and significantly decrease the waiting time with MATLAB!

Function Description:

Write a function that takes in two inputs: an array representing all the current orders at Panera and a name to search for in the order list. Your function should output the number of orders remaining until the specified order is up.

Example:
```
orders =    'Juliana '        rem = panera(orders, 'Chima')
            'Amanda  '
            'Chima   '        rem => 2
            'Jake    '
            'Amarachi'
```

Notes:
- The input array will always have as many columns as there are letters in the longest name.
- The name you are looking for is guaranteed to appear exactly once.
- You should **NOT** use iteration for this problem.

Hints:
- Remember that the `strfind()` function only takes row vectors as inputs.
- Think about what different operations, like linearizing and transposing, do to the elements of an array.

Function Name: `starbucks`

Inputs:

1. (*double*) An Nx6 array representing the menu
2. (*double*) A 1x6 array representing the secret menu item

Outputs:

1. *(double*) An Nx2 array representing the updated "grande" portion of the menu

Background:

On Monday at 2:45 PM sharp, you go to the CULC Starbucks to beat the lines and grab a coffee. Unfortunately, the line has filled the entire second floor, snaking around the stairs and all the way to the CS 1371 help desk in room 272. You quickly pull up a menu of the prices and calorie counts of all the drink options and decide to whip up a frothy MATLAB function to help you decide what to get from Starbs!

Function Description:

Each pair of columns in the input array represents a drink size: tall, grande, or venti, respectively. The first column of the pair will contain a price for the drink, and the second will contain the calories in that drink. Each row represents a different drink. You'll make the following changes to the menu:

- Make the menu more healthy by removing the entire row of the drink with the highest number of calories in the venti size.
- Append the price and calorie statistics for your favorite secret menu item, (the unicorn frappuccino!), which is the second input, to the bottom of the menu.
- Narrow down your options to only the two columns for grande price and calories.
- Calculate a ratio of price per calorie and sort the rows of the grande menu based on this, so that the ratio is increasing as you go down the columns.

Example:

```
                    (Tall)    (Grande)    (Venti)
>> menu    =  [ 2.85 150 3.45 190 3.95 250
                3.25 190 3.95 240 4.25 310
                1.95  60 2.45  80 2.95 120 ]
>> secret  =  [ 3.75 350 4.45 420 4.95 540 ]
>> newMenu = starbucks(menu, secret)
>> newMenu => [ 4.45  420
                3.45  190
                2.45   80 ]
```

Notes:

- Overpriced and caffeinated beverages are not encouraged as a way to gain nutritional value.