Function Name: `grammarCheck`

Inputs:

1. *(char)* A simple sentence

Outputs:

1. *(logical)* A value indicating whether the sentence is grammatically correct

Function Description:

As a Georgia Tech student taking CS 1371, you are likely finished or nearly finished with your English education. With a lack of courses keeping your grammar up to par, you decide to harness the power of MATLAB to check the grammatical correctness of your sentences! Write a function that takes in a simple sentence as a string and outputs a logical value stating whether the sentence has correct grammar. At this point in the course, it will be necessary to check only two components of the sentence to ensure that it is grammatically correct:

1. The sentence must begin with a capital letter.
2. The sentence must end with a period (`'.'`), a question mark (`'?'`), or an exclamation point (`'!'`).

If the input sentence meets both of these requirements, the function should return true. Otherwise, it should return false.

Notes:

- The input string will not be empty.
- Even an input sentence like `'HeLLO...'` should return true.
- This function should not be used to proof your resume.

Function Name: `lottery`

Inputs:

1. *(char)* A 1xN string representing a bought lottery ticket number
2. *(char)* A 1xN string representing a winning lottery ticket number
3. (*double)* The maximum prize amount

Outputs:

1. *(double*) A double representing how much money you've won

Background Information:

PowerBall fever has gripped the nation over the past few weeks, and although the jackpot has already been claimed, you are inspired to pursue your own chance at winning money. You buy a single lottery ticket, hoping to land a big prize.

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 formatted as follows:

- They are composed of 6 positive integers separated by dashes.
- There will be no extraneous characters in the string.

You will also be given the maximum prize amount (excluding the PowerBall bonus) of the lottery which you will use to determine the winnings.

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 20% of the maximum prize to your total prize.
2. If the last (6th) number matches, then double the prize amount determined by the first 5 numbers.
3. None of the first 5 numbers in your ticket number will match the winning lottery ticket's PowerBall number.

For example:

Bought lottery ticket	→	`'13-45-33-19-29-8'`
Winning lottery ticket	→	`'29-10-6-13-41-8'`
Maximum prize amount	→	`100,000`

The matching digits are 29, 13, and 8. Since 29 and 13 are both in the first five digits of each ticket number, each contribute 20% of the total. So the total is $20,000 + $20,000, or $40,000. In addition, since the last digit, 8, matches between the bought and winning tickets, the overall prize amount is doubled to $80,000.

You should round the final amount to the nearest cent.

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).

Hints:

- The `strfind()` function will be useful.

Function Name: `caesarSalad`

Inputs:

1. *(char)* A 1xN string of a single, lowercase word
2. *(double)* An integer describing the shift, or the “shift number”

Outputs:

1. *(char)* The input word encoded using the Caesar cipher

Caesar Cipher Information:

The Caesar cipher is named after Julius Caesar, who, according to Suetonius, used it with a shift of three to protect messages of military significance. It is unknown how effective the Caesar cipher was at the time, but it is likely to have been reasonably secure because most of Caesar's enemies would have been illiterate and others would have assumed that the messages were written in an unknown foreign language.

An example of a Caesar shift is the ROT13 algorithm, a simple method often used to obscure text such as joke punchlines and spoilers online.

Function Description:

In a Caesar cipher, each letter of a word is shifted by a specified amount. For example, if the shift is 3, then the letter `'a'` would be coded as the letter `'d'`. These shifts also wrap around the end of the alphabet. So the letter `'y'` shifted by 4 is `'c'`. Write a function that takes in a string of a single word and uses the Caesar cipher with the input shift number to encode it. Only lower case letters will be included in the input string; any other characters—such as spaces, periods, commas, etc.—will not be included as part of the input.

Notes:

- The function should work for both positive and negative integer shift values
- There is no limit to the value of the shift number in the second input.

Hints:

- The `mod()` function will be useful.

Function Name: `criminalMinds`

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 is lying

Function Description:

After years of reading Nancy Drew and watching Bones, you realize your true passion lies in criminal justice. After years of training with the FBI, you are finally working a case, and you have four suspects—only one of which is the criminal. You give each one a polygraph 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. Since you were a pro at MATLAB back in the day, you decide to write code to assist you in finding the criminal.

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, find which of the four suspects is lying, and, thus, is the criminal.

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

Notes:

- The suspect who is the liar 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.