Function Name: `leafPile`

Inputs:

1. (*char*) A string representing all of the leaves
2. *(char)* A color of leaf to search for

Outputs:

1. *(double*) The number of leaves of that specific color

Background:

Fall is in the air. This means that means leaves are on the ground! You walk outside and see thousands of leaves blanketing campus and immediately want to count the number of leaves of a specific color there are. Assuming one of your 1371 TAs has meticulously documented the color of every leaf on campus, use your new MATLAB skills to find the number of leaves of that specific color in the pile.

Function Description:

Write a function that takes in a character vector containing the colors of all the leaves in a pile as the first input, and a single color as the second input. Find the number of times that the specified color occurs in the first input. The search should be **case insensitive**, meaning that you should ignore whether letters or words are uppercase or lowercase when searching through the string.

Example:

```
>>pile = 'red brOwn green yellow BrOwN MATLAB GrEEn rEd purple Blue brown'
>>color = 'brown';
>>num = leafPile(pile, color)
      ==> num = 3
```

Notes:

- The function should **not** be case sensitive. So ‘RED’ and ‘red’ should both count if the second input is ‘Red’

Hints:

- The function strfind() will be very useful

Function Name: `fallingIntoThemDMs`

Inputs:

1. (*char*) An encoded character vector ending in a space and a number

Outputs:

1. *(char*) The decoded message

Background:

Fall is in the air. The leaves are changing colors, the temperature is dropping, Yellow Jacket fans are packing into Bobby Dodd, and you just so happen to be taking CS 1371 at the same time as your crush. Now, in a surprise turn of events, your crush has fallen into your DMs- in the middle of class! However, they sent you a secret code that you will have to use your MATLAB skills to decipher.

Function Description:

The message will consist of a string of ASCII characters followed by a space, then a number. To reveal the message, you must get the number from the end of the string and add its value to each of the other characters in the string. In essence, you are manipulating the ASCII values and therefore changing the string to reveal the message!

Example:

Given `'B_ffi 6',` your function should output `'Hello'`

Notes:

- There will only be 1 space in the input string
- The number at the end can be any number of digits
- Make sure to remove the space and number from your output

Hints:

- The `strtok` and `str2num` functions may be useful
- Don't forget about casting

Function Name: `knitted`

Inputs:

2. *(double)* 1xM vector to be inserted in the odd position indices
3. *(double)* 1xN vector to be inserted in the even position indices

Outputs:

2. *(double)* 1xQ vector with input vectors “knitted”

Background:

Fall is in the air, so your new wardrobe will no longer do. It’s time to trade in all your free FASET t-shirts for some handmade sweaters! Your job is to use MATLAB to knit together two input vectors.

Function Description:

Given two vectors, reverse the first input vector, sort the second, then output a larger vector where the values in the first vector are inserted in the odd indices and the values in the second vector are inserted in the even indices. If the two vectors are not the same length, the shorter vector should be extended by adding 1s to the end.

Example:

```
sweater = knitted( [83, 67, 1, 7, 3, 1], [3, 4, 1, 2])
sweater ---> [1 1 3 2 7 3 1 4 67 1 83 1]
```

Function Name: `getPumpkinPi`

Inputs:

1. *(double)* Number of terms to use in approximation

Outputs:

1. *(double)* Result of the approximation

Background:

Fall is in the air, which means that it's time for pumpkin! While your friends practice their pumpkin carving skills and drink some pumpkin spice lattes, you decide to bake a pumpkin pie. You want your pie to be perfectly circular, of course, and for that you need to calculate π!

Function Description:

There are many ways to calculate the mathematical constant π. One of the ways to do it is based on the series expansion of the inverse tangent function. This fact will allow you to calculate π:

$$\tan^{-1}(1) = \sum_{n=1}^{\infty}(-1)^{(n-1)}\frac{x^{2n-1}}{2n-1} = 1 - \frac{1}{3} + \frac{1}{5} - \ldots = \frac{\pi}{4}$$

This equation relates $\frac{\pi}{4}$ to the sum of $\frac{1}{n}$ for every other odd integer n starting with 3 (so 3,7,11,etc.) subtracted from the sum of $\frac{1}{n}$ for every other odd integer n starting with 1 (so 1,5,9,etc). Write a function that approximates π using this sum, with the number of terms specified by the input. Round your answer to the sixth decimal place.

For example, if the input was 5, your answer should be:

$$4 * (1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9}) = 3.339683$$

Notes:

- The input is guaranteed to be a positive integer greater than 1.
- This can and should be done entirely using vector math.
- Remember that putting a dot before an operation (`./, .*, .^`) performs the operation element-by-element on a vector.
- Do not attempt to simply output MATLAB's built in `pi` constant; it will not work.

Hints:

- Try typing the namesake of this series into the function for a fun easter egg!

Function Name: `fallingStocks`

Inputs:

1. (*double*) 1xN Vector of initial stock prices
2. (*double*) 1xN Vector of projected final stock prices
3. (*double*) Total amount of money to invest

Outputs:

1. *(double*) Vector of money that could be made by buying each stock

Background:

Fall is in the air, and coincidentally the stock prices have also been falling for a while. However, you're a pro investor, so you know that the stocks are about to rebound. You think it's time to invest in some stocks so you can make some money, and you turn to MATLAB to figure out how much money you can make!

Function Description:

Given a vector describing initial stock prices and a vector of the predicted final stock prices, as well as how much money you have to invest, you will figure out the best stocks, in order, to spend your money on

- Calculate the amount of money you could make by purchasing a stock. Do this for each stock and store these values in a vector.
- Calculate the number of each type of stock you are able to buy with the money you have using their initial prices. Assume you can only buy a whole number of stocks.
- Use these values to figure out the profit you can make by spending as much of your money on each type of stock.
- Sort the output vector from the greatest amount of money that could be made to the least.

Example:

```
fallingStocks([20,30,40], [30,60,80], 100) => [90, 80, 50]
```

Notes:

- Round the output vector to the third decimal place.
- CS 1371 TAs are not liable for any losses incurred on the stock market as a result of this problem.
- It is okay if some of the values end up negative.

Hints:

- The `'descend'` option for `sort()` may be useful.