Drill Problem #1

Function Name: deepestLayer

Inputs:

1. (*cell or other*) A 1x1 nested cell

Outputs:

1. (*char, logical, or double*) Contents inside the input cell

Function Description:

A nested cell is a cell that contains another cell within itself; it can be nested a multiple number of times (cell-ception). Write a MATLAB function that will take an input, 1x1 nested cell and output the contents inside the innermost cell, through recursion.

For example, if the input cell, `ca`, were the following:

```
ca = {{{{{{{22894}}}}}}}
```

then the function would recursively peel each layer of the cell away as follows:

call 1: `ca = {{{{{{22894}}}}}}`
call 2: `ca =  {{{{{22894}}}}}`
call 3: `ca =   {{{{22894}}}}`
…
…
…
call n (where n=7 in this example):

```
ca =  22894
```

Which would be the output value.

Notes:

- If the input is not of class `'cell'` just output the input as itself.
- You will not have a cell with a length greater than one.
- You **MUST** use recursion to solve this problem.

Drill Problem #2

Function Name: collatz

Inputs:

1. (*double*) Any positive integer

Outputs:

1. (*double*) The number resulting from the algorithm
2. *(double)* The number of recursive steps required by the algorithm

Function Description:

The Collatz conjecture (also known as the 3n+1 conjecture) was proposed by Lothar Collatz in 1937. The conjecture says that any positive integer n can be recursively manipulated **to be a number less than 2** by the following algorithm:

- if the number is even, divide it by 2
- if the number is odd, multiply it by 3 and add 1
- repeat (i.e. recursively call the function)

Your job is to write a recursive MATLAB function that implements the Collatz conjecture. Your function should output the final number that the algorithm reaches as well as the number of recursive calls it took to get there.

Notes:

- If you try running this function with very large numbers (as in 40-digit numbers), MATLAB will crash as it will reach its maximum recursion limit.

Hints:

- You may find it useful to make a helper function to keep track of the number of recursive calls that have been made.

Drill Problem #3

Function Name: speedStack

Inputs:

1. (*double*) The length of the base of the pyramid
2. (char) A pyramid character

Outputs:

1. (*char*) The created pyramid of characters

Function Description:

Speed Stack is a competition in which contestants stack cups in a pyramid conformation with the goal being to do so in the shortest amount of time. Since not everyone in the world has been blessed with great manual dexterity, your local Speed Stack organizers have decided to hold an alternate competition in which a pyramid with a size of their choosing must be created in the shortest amount of time possible by any means the contestants' desire. Recognizing that using a computer is an almost surefire way to win, you decide to enter the competition with MATLAB.

Write a recursive function that takes in the length of the base of the pyramid as well as any single character with which to build a pyramid (pyramid character) and outputs an array of class `'char'` that contains the pyramid. The dimensions of your array should be <length of base> by <(2*length of base)-1>. Starting in row 1 of the array, each subsequent row should have 1 more pyramid character than the row above it and a single space (ASCII value 32) should separate each pyramid character within a row. Any other index within the array that does not contain a pyramid character should be filled with a space (ASCII value 32).

For example, for a pyramid of base length 10 and constructed from the character `'/'`, your function should output a 10x19 character array as shown below (in order to better visualize the actual construction of the array, the ASCII values for this array have been printed below with the pyramid character's value highlighted)

```
                  /
                 / /
                / / /
               / / / /
              / / / / /
             / / / / / /
            / / / / / / /
           / / / / / / / /
          / / / / / / / / /
         / / / / / / / / / /
```

Drill Problem #3

32	32	32	32	32	32	32	32	32	47	32	32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32	47	32	47	32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	47	32	47	32	47	32	32	32	32	32	32	32
32	32	32	32	32	32	47	32	47	32	47	32	47	32	32	32	32	32	32
32	32	32	32	32	47	32	47	32	47	32	47	32	47	32	32	32	32	32
32	32	32	32	47	32	47	32	47	32	47	32	47	32	47	32	32	32	32
32	32	32	47	32	47	32	47	32	47	32	47	32	47	32	47	32	32	32
32	32	47	32	47	32	47	32	47	32	47	32	47	32	47	32	47	32	32
32	47	32	47	32	47	32	47	32	47	32	47	32	47	32	47	32	47	32
47	32	47	32	47	32	47	32	47	32	47	32	47	32	47	32	47	32	47

Notes:

- You must use recursion to solve this problem. Any function not using recursion will result in a 0 for this problem.
- Your function should work for any base length that is <498, since this is MATLAB's maximum recursion limit, allowing for helper functions.
- The pyramid character is guaranteed to be of length 1, and the size of the pyramid base is guaranteed to be at least 1.

Drill Problem #4

Function Name: r_nFib

Inputs:

1. (*double*) A number to begin the sequence
2. (*double*) An positive integer (N) denoting the number of terms to return

Outputs:

1. (*double*) A 1xN vector of the resulting Fibonacci sequence

Function Description:

Remember the `nFib` problem couple weeks ago?! It made its way back because it's SO important! In case your memory need a little refreshing:

The Fibonacci sequence is very important in mathematics, physics, nature, life, etc. Each number in the sequence is the sum of the previous two values, where the first two numbers in the sequence are always 0 and 1, respectively.

Write a MATLAB function that puts a twist on the classic Fibonacci sequence. This function will input a number to begin the sequence and the number of terms of the Fibonacci sequence to evaluate, and it will output a vector of the corresponding sequence. If the initial term is a 0 or 1, the second term will be a 1; if the initial term is any other number, the second term will be that initial number, repeated.

Therefore, one can find the sequence out to 6 terms, beginning at the number 2, with the following:

$$Fibonacci_1 = 2$$
$$Fibonacci_2 = 2$$
$$Fibonacci_3 = Fibonacci_1 + Fibonacci_2 = 4$$
$$Fibonacci_4 = Fibonacci_2 + Fibonacci_3 = 6$$
$$Fibonacci_5 = Fibonacci_3 + Fibonacci_4 = 10$$
$$Fibonacci_6 = Fibonacci_4 + Fibonacci_5 = 16$$

Meaning the output 1xN vector of the entire sequence to 6 terms would be:

```
out =  [2 2 4 6 10 16]
```

Your job is to write a new MATLAB function, `r_nFib`, which solves the `nFib` problem recursively!

Notes:

- You will not have any negative input values
- You **MUST** use recursion to receive credit for this problem.
- As with `nFib`, for an input requesting zero terms, the output should be a 1-by-0 Empty Matrix.

Hints:

- You may find a helper function useful.

Drill Problem #5

Function Name: sixDegreesOfWaldo

Inputs:

1. (*struct*) 1x1 Structure representing a person

Outputs:

1. (*logical*) Whether Waldo can be found
2. (*double*) The degrees of separation from the starting person to Waldo

Function Description:

After spending your childhood years desperately scanning page after page for the Hide-And-Seek World Champion (also known as the infamous Waldo), you decide enough is enough and you are going to use MATLAB to help you decide once and for all if Waldo is hiding among us.

Write a function called `sixDegreesOfWaldo` that takes in a 1x1 structure representing a single person. The input structure is guaranteed to have the following fields:

1. `Name`: (String representing the name)
2. `Age`: (double representing the age)
3. `Friends`: (1xN Structure array representing all of this person's friends)

Note that for each friend, they are guaranteed to have the above 3 fields, but may have any number of extra fields in addition to the 3 above.

Your job is to recursively search through all of the people available by checking all of the first person's friends, then each of their friends, and so on.

The first output of the function will be a single logical indicating if Waldo is a friend of anyone in the group (e.g. if Waldo is present). Unfortunately, Waldo is sneaky and doesn't always put `'Waldo'` in the Name field, so **the first output of the function should be true if ANY field of ANY of the people in the group contains the string `'Waldo'`.** Note that the spelling and case must appear exactly as shown.

The second output of the function will be a double representing the degrees of separation from the first person to Waldo. **If Waldo does not exist in the group, this output should be 0. The degrees of separation is the number of levels of friends you have to move through (from the beginning person) to find Waldo.** If the first person is Waldo, then the degrees of separation will be 0, if (instead) a friend of theirs is Waldo, the degrees of separation will be 1, if (instead) a friend of a friend of the first person is Waldo, the degrees of separation will be 2, and so on.

Notes:

- There will never be more than 1 Waldo in the group.
- Do not hardcode the field names for the friends: there is no guarantee how many there will be, nor what they will be called.
- You **must use recursion** to solve the problem**.**

Drill Problem #5

Hints:

- Think about the best way to keep track of the results of each friend as you traverse the group.
- A Visual example is included below.

Visual Example:

- The following is a possible input to the function:

```
person =

       Name: 'Otis'
        Age: 20
    Friends: [1x9 struct]
```

- This person has 9 total friends. The first of these friends looks like this:

```
K>> person.Friends(1)

ans =

                                Name: 'Florentina'
                                 Age: 98
                             Friends: []
                             Student: 1
    Doughnuts_eaten_in_the_last_week: 66
```

- The above person has no friends in the list. Since they are not Waldo, we are done searching for this friend. The 8th friend of our original person looks like this:

```
K>> person.Friends(8)

ans =

                              Name: 'Almeda'
                               Age: 44
                           Friends: [1x8 struct]
                           Student: 1
    Doughnuts_eaten_in_the_last_week: 6
```

- The above person has 8 friends, each of which need to be searched if they are (or have any friends who are) Waldo. Now let's look at Almeda's 4th friend (Almeda is the person above):

```
K>> person.Friends(8).Friends(4)

ans =

                              Name: 'Vito'
                               Age: 10
                           Friends: [1x10 struct]
                           Student: 1
                Matlab_Lines_Written: 'Waldo'
    Doughnuts_eaten_in_the_last_week: 61
```

- The `'Matlab_Lines_Written'` field has a value of `'Waldo'` which means this person is Waldo in disguise. In this case, the first output of our function would be `true`.
- The second output of our function would be `2` since we had to go through two layers of friends to reach Waldo (The first layer was Almeda, the second layer was Waldo himself – since he is a friend of Almeda).

Drill Problem #5

Extra Help Provided:

Function Name: generatePeople (already written)

Inputs:
None

Outputs:
1. (struct) Structure representing a person

In order to help you test your function (`sixDegreesOfWaldo`), we have provided the following code for you as a .p file. This function generates a possible input for `sixDegreesOfWaldo`. Note that the generation is completely random – it is only made to help you test your code. *It does not guarantee any kind of score after grading*.

To use it, simply run the function. There are no inputs, and only one output – a sample input to the sixDegreesOfWaldo function. You can use the output of this function as an input to both the solution file and your own code to help you test your `sixDegreesOfWaldo` code.

These are in supplement to the test cases already provided in the hw11.m file.

Note:

- **Do not** include the `generatePeople` function in your `sixDegreesOfWaldo` function. ***It will result in a score of 0 for your entire homework 11 assignment***.