Extra Credit

Function Name: `solveSudoku`

Inputs:

1. (*double*) 9x9 array representing an unsolved Sudoku board

Outputs:

1. (*double*) 9x9 array representing a solved Sudoku board

Function Description:

After having a very enlightening, yet slightly awkward interaction with Luna Lovegood and her father, they encourage you to read The Quibbler, the wizarding tabloid. You try your best to understand the odd articles about conspiracy theories and imaginary creatures, but as a muggle, you are at a complete loss. Fortunately, there are some fun mind games at the back of the publication, one of which is sudoku! In the fine print, you read that magic is recommended to solve this "challenging puzzle". While you might not have magic, you have the next best thing, MATLAB! (which might as well be magic)

Write a recursive function that takes in an array representing a Sudoku board and solves it. The three rules for filling in each box are:

1) Each row may not have repeating numbers
2) Each column may not have repeating numbers
3) Each 3x3 block may not have repeating numbers

You can find more detailed and visual instructions here, and it is **strongly recommended** that you follow the link if you are not familiar with sudoku. The empty boxes are represented by NaN's in the array.

Notes:

- Each board will only have one unique solution.
- The numbers will always be 1:9.

Hints:

- Make sure to use isnan() to check for the NaN's.
- The beauty of computers is that they are willing and able to solve problems by brute force methods that a sane human would never want to try.

Extra Credit

Function Name: `spaceJam`

Inputs:

1. (*struct*) 1x1 structure representing a computer directory

Outputs:

1. (*double*) The total amount of space taken up by all files in that directory and any subdirectories
2. (*char* **or** *cell*) The path(s) of the file(s) that consumes most space

Function Description:

Can you believe that your smart phone has 10 times the storage capacity than a hard drive sold in 2000? Hard drives were compared based on how many megabytes of information they can retain. In the olden times, we would have had to make sure our computer had enough space to install the latest version of MATLAB so we can do CS 1371 Homework!!

Of course, as a wizard, this means absolutely nothing to you, so you are going to have to depend on a muggle computer to learn about muggle computers! To help with this Catch-22 scenario*, write a MATLAB function which takes in a 1x1 structure, with each field corresponding to a file or subdirectory name, and the value of each field being either a double or another 1x1 structure. If the value of a field is of type double, then the field name is a file name, and that value is the space that file takes up. Otherwise, that field name will be the name of a subdirectory, and the value will be a structure that contains additional files or subdirectories.

First, sum up the space taken by all files in the directory/subdirectories and round this value **up** to the nearest integer. You can assume that an empty subdirectory takes up no space. Then, output the path(s) of the file(s) that consumes the most space, starting with a "\", and then each subdirectory name and ultimately file name separated by a "\". For example:

```
\subdirectory1\subdirectory2\largestfile
Or
\largestfile
```

If there is only one largest file, output the path as a string (type char). Otherwise, if there are more than one file with the same size that is the largest, output each path as a string as part of a cell array of strings, sorted alphabetically.

Notes:

- *It is a Catch-22 because we have to write a MATLAB function in order to determine if we can install MATLAB in order to do CS homework/write MATLAB functions.
- There is no guarantee as to the degree of nesting structures can have

Extra Credit

Function Name: `sixDegreesOfPotter`

Inputs:

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

Outputs:

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

Function Description:

You may be familiar with the game "Where's Waldo?" after spending many years scanning page after page in search for the Hide-And-Seek World Champion, but are you familiar with the Wizarding World's famous game "Where's Potter?"? Many wizards spend their early years trying to find The Boy Who Lived in the midst of many other wizards. After many attempts to find Harry Potter, you decide enough is enough and that you are going to use MATLAB to help you decide once and for all if Harry Potter can be found among us.

Write a function called `sixDegreesOfPotter` 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 Harry Potter is a friend of anyone in the group (e.g. if Harry Potter is present). Unfortunately, Harry Potter is sneaky and doesn't always put `'Harry Potter'` 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** `'Harry Potter'`**.** 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 Harry Potter. **If Harry Potter 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 Harry Potter.** If the first person is Harry Potter, then the degrees of separation will be 0, if (instead) a friend of theirs is Harry Potter, the degrees of separation will be 1, if (instead) a friend of a friend of the first person is Harry Potter, the degrees of separation will be 2, and so on.

Notes:

- There will never be more than 1 Harry Potter 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.

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: 23
      Friends: [1x9 struct]
```

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

```
>> person.Friends(1)
ans =
                          Name: 'What's Her Face'
                           Age: 21
                       Friends: []
                      Students: 20
      Doughnuts_eaten_last_week: 66
```

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

```
>> person.Friends(8)
ans =
                              Name: 'The Professional MATLAB'
                               Age: 'Unknown'
                           Friends: [1x8 struct]
                          Students: 0
      Global_issues_solved_last_week: 66
```

The above person has 8 friends, each of which need to be searched if they are (or have any friends who are) Harry Potter. Now let's look at The Professional MATLAB's 4th friend:

```
>> person.Friends(8).Friends(4)
ans =
                     Name: 'The Anger Translator'
                      Age: 22
                  Friends: [1x10 struct]
      MATLAB_Lines_Written: 'Harry Potter'
```

The `'Matlab_Lines_Written'` field has a value of `'Harry Potter'` which means this person is Harry Potter 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 Harry Potter (The first layer was The Professional MATLAB, the second layer was Harry Potter himself – since he is a friend of The Professional MATLAB).

Extra Help Provided (for sixDegreesOfPotter)

Function Name: `generatePeople` (already written)

Inputs:

none

Outputs:

1. (*struct*) Structure representing a person

In order to help you test your function (`sixDegreesOfPotter`), we have provided the following code for you as a .p file. This function generates a possible input for `sixDegreesOfPotter`. 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 sixDegreesOfPotter function. You can run the output structure in the provided solution file and your own code to test the accuracy of your code. These are in supplement to the test cases already provided.

Note:

- **DO NOT** include the `generatePeople` function in your `sixDegreesOfPotter` function. ***It will result in a score of 0 for your entire homework 12 assignment***.