Function Name: `tomAndJerry`

Inputs:

1. (*struct*) A structure representing the old Tom
2. (*struct*) A structure representing new aspects of Tom
3. *(char)* A field of Tom to remove

Outputs:

1. *(struct*) A structure representing the new Tom

Background:

Tom has struggled to capture Jerry for years, so Tom decides to go on a long introspective journey to the core *structure* of his being.

Function Description:

You will be given three inputs. The first one will be a structure representing Tom. The second one will be a structure representing the aspects of himself he wants to change. The third will be a character vector of a field to remove.

Write a function that goes through the fields of the second structure and uses them to change the values of the same field names of the first structure. Then remove the field named by the third input from the first structure, and output the edited first structure.

Example:

```
>> tom =
    Cleverness: 'Kinda'
    Gullibility: 'Very'
    Cuteness: 'Ehhhh'
    Height: 'Not tall'

>> new =
    Cleverness: 'Extreme'
    Cuteness: 'Quite'

>> newTom = tomAndJerry(tom, new, 'Gullibility')

newTom =
    Cleverness: 'Extreme'
    Cuteness: 'Quite'
    Height: 'Not tall'
```

Notes:

- The fields of the second input are guaranteed to be fields of the first input, but the first input may have fields that do not exist in the second input.
- The fieldname contained in the third input will always be a field of the first input.

Function Name: `garfield`

Inputs:

1. (*struct*) A 1xN structure array containing information about cats

Outputs:

1. *(struct*) A 1xM updated structure array with no occurrences of Monday

Background:

Garfield is a grumpy cat. You think that you could make him happier by setting him up with a lady cat friend. The only problem is that you know how much he hates Mondays, and that any mention of the word will set him off, and your plan will fail. Luckily, you have MATLAB to ensure that all of the lovely candidates are safe to meet Garfield.

Function Description:

Write a function that iterates through the fields of a structure vector. If any of the fields within a structure in the array contain the value `'Monday'`, regardless of case, then entire structure should be removed.

Example:

```
>> struct =
```

`Name:'Monday'`
`Color:'Black'`
`faveWord:'Lasagna'`
	`Name:'Lucy'`
	`Color:'Gray'`
	`faveWord:'No'`
		`Name:'Princess'`
		`Color:'White'`
		`faveWord:'monday'`
			`Name:'Crookshanks'`
			`Color:'Orange'`
			`faveWord:'Magic'`

```
>> newStruct = garfield(struct)

newStruct =
```

`Name:'Lucy'`
`Color:'Gray'`
`faveWord:'No'`
	`Name:'Crookshanks'`
	`Color:'Orange'`
	`faveWord:'Magic'`

Notes:

- The function should be case insensitive - it should delete a structure containing `'monday'` regardless of capitalization.
- The word `'monday'` will NOT be a substring of another value.

Function Name: `birdNest`

Inputs:

1. (*struct*) A 1x1 structure representing a classroom

Outputs:

1. *(char*) A sentence describing where the bird was found

Background:

Evil forces are at work in the Instructional Center. The ancient runes etched into the second floor bathroom stalls foresaw the nightmare that was about to awaken, but the masses of Georgia Tech students were powerless against the terror that was about to be unleashed.

One dark and stormy Thursday afternoon, the unspeakable finally happened. A malicious presence rose from the bottomless pit beneath the IC and began to wreak havoc upon the innocents. This monstrosity has had many names; most have been lost to antiquity, and now the abomination is known only as the infamous IC bird. Now, it's up to Georgia Tech's warrior cat population to stop the IC bird's murderous rampage.

Function Description:

Write a function that takes in a 1x1 structure representing a classroom. The structure will have the following fields:

`RoomNumber`
`NumberSeats`
`ContainsBird`
`Exit`

`RoomNumber` is a double that represents the room number of the classroom, `NumberSeats` is a double that represents the number of seats in the classroom, and `ContainsBird` is a logical representing whether or not the bird was found in the room. The fourth field, `Exit`, represents what lies beyond the door leading out of the classroom. This can either contain **any** string such as `'outside'` or it can be another 1x1 structure representing an adjacent classroom.

Your task is to search for the bird by traveling along adjacent classrooms. If the bird is not found in the first classroom and the `Exit` field contains another room structure, go on to the next room and repeat the process until either the bird is found or there are no more classrooms to search. If you find the bird, output the following string:

```
'After an epic chase spanning <number of rooms searched> rooms, the cats
                found the bird in IC <room number>!'
```

If you do not find the bird and run out of rooms to search, output the following string:

```
'It was a stunning chase, but the bird escaped.'
```

Continued...

Example:

```
>> roomStruct =

RoomNumber: 103
NumberSeats: 150
ContainsBird: false
Exit: [1x1 struct]

>> roomStruct.Exit =

RoomNumber: 115
NumberSeats: 30
ContainsBird: true
Exit: 'outside'

>> description = birdNest(roomStruct)

description = 'After an epic chase spanning 2 rooms, the cats found the bird
in IC 115!'
```

Function Name: `schrodingersCat`

Inputs:

1. (*struct*) A MxN structure array
2. (*double*) A 1x2 vector of your starting position

Outputs:

1. (*char*) A statement describing the cat, its location, and how many steps it took

Background:

In 1935, Erwin Schrödinger proposed the famous thought experiment in which cat in a box is simultaneously alive and dead. Only once the box is opened can the cat's state of being be determined. Using your MATLAB skills, you have set up the experiment yourself, and you want to figure out if your cat is alive or dead. However, you set up a large array of many boxes, and forgot which one the cat is in. Good thing you have MATLAB to help you out!

Function Description:

You are given a structure array and a set of indices within the structure array at which to start. Each structure in the array contains only one field, called `next`. The `next` field of each structure contains either a 1x2 vector of the indices of the next structure to search or a character vector describing the state of the cat. The character vector describing the cat will either be `'alive cat'` or `'dead cat'`.

To find the cat, first check the structure at the index given by the second input. Next, check the structure at the indices given in the `next` field of the previous structure you checked. This process should continue until you encounter the cat, whether it is dead or alive. Be sure to keep a count of how many steps you have gone through before you reach the cat. Your output statement should have the following form:

```
'The <alive or dead cat> was found at position (<row>,<column>)
                    after <num steps> steps.'
```

Notes:

- The path from the given starting position is guaranteed to reach the cat.

Function Name: `warriors`

Inputs:

1. (*struct*) An 1xN structure array
2. (*char*) A field name

Outputs:

1. *(struct*) An 1xM updated structure array
2. (*char*) A descriptive statement about the clan battle

Background:

While reading everyone's favorite childhood book series about clans of feral cats, *Warriors*, you realize that the plot is quite difficult to keep up with at times. Rather than simply following the exciting adventures of the protagonist, Firepaw of the ThunderClan, like a casual reader, you decide to go the extra step. Since you are such a dedicated fan of this incredible series, you will write a MATLAB function to track the battles of the clans throughout the saga.

Function Description:

Write a function that takes in an 1xN structure array and a field name of a statistic that is the most important in a particular battle. Each structure in the array represents a different clan of cats. Based on the values of the field from the given field name (second input) which are guaranteed to be of type double, you will determine which clans battle and who will win. The clan with the highest value in the given field will win in a warrior battle against the clan with the lowest value in the given field. Any intermediary clans will not be affected by the battle. Make the following updates to the structure array based on this battle:

1. The winning clan's structure's value in the field given by the second input should be doubled.
2. The winning clan's structure's value of the `Territories` field, represented by a cell array of strings, should be updated to include the values of the `Territories` field in the losing clan's structure.
3. The losing clan's value in the field given by the second input should be set to zero.
4. The losing clan's value for the `Territories` field should become an **empty cell array**, `{}`.
5. Sort the entire array in descending order based on the values in the field given by the second input. Do this using the `'descend'` input to sort.

In addition to updating the structure array, the function should also output a descriptive statement in the following format:

```
'Following the warrior code, fearless leader <winning leader> led
    <winning clan> to victory against <losing clan>.'
```

Continued...

Example:

```
>> clans =
```

Name : 'ShadowClan' Leader : 'Brokenstar' Territories : {'northeast marshes'} Strength : 75	Name : 'ThunderClan' Leader : 'Firestar' Territories : {'southeast woodlands'} Strength : 80

```
>> [updatedClans, result] = warriors(clans, 'Strength')

updatedClans =
```

Name : 'ThunderClan' Leader : 'Firestar' Territories : {'southeast woodlands', northeast marshes} Strength : 160	Name : 'ShadowClan' Leader : 'Brokenstar' Territories : {} Strength : 0

```
result = 'Following the warrior code, fearless leader Firestar led Thunderclan to victory against Shadowclan.'
```

Notes:

- The input structure array is guaranteed to have the following fields: `'Name'`, `'Leader'`, and `'Territories'`.
- It is guaranteed that the values in the field given by the second input will be a 1x1 double.
- It is guaranteed that the value in the `Territories` field will be a cell array.
- There will not be any ties.