Function Name: `animalHotel`

Inputs:

1. (*cell*) An MxN cell array

Outputs:

1. *(struct*) A 1x(M-1) structure array

Background:

You are the manager of the World Class Animal Hotel and have been tasked with keeping a record of all the guests that stay in the hotel! On your first night, you get a little flustered by the pace of the job and quickly jot down all of the guests that come in and stay in the hotel in a cell array. Unfortunately, your boss informs you that all information must be converted to a structure array before it can be filed away for safekeeping.

Function Description:

Write a function that takes in a cell array and stores the given information in a structure array. The first row of the cell array contains the field names you will want in your structure array. Each subsequent row will contain information about another guest, in order corresponding to the field names in row 1. For each row after the field name, create a structure that contains all the given information.

Example:

```
cellArr = {'Name', 'Species', 'FavoriteFood'; 'Garfield', 'Cat', 'Tuna'; 'Patches' , 'Dog', 'Steak'; 'George', 'Monkey', 'Bananas'}

StructArr = Name:          Garfield    Patches     George
            Species:       Cat         Dog         Monkey
            FavoriteFood:  Tuna        Steak       Bananas
```

Notes:

- Since the first row of the cell array is field names, the length of the structure array should be 1 less than the number of rows in the cell array.
- There will always be at least two rows in the input cell array.

Function Name: `remodelZoo`

Inputs:

1. (*struct*) An MxN structure array representing a zoo

Outputs:

1. *(struct)* An MxN structure array representing the remodeled zoo

Background:

The Atlanta Zoo has made a terrible mistake. Revenues were low, so they decided to restructure the zoo in order to improve people's experience! However, they decided to hire a u[sic]GA grad to do the remodeling, and he did a terrible job, making things worse! Luckily, Georgia Tech is right around the bend, and you can use your MATLAB skills to help them out!

Function Description:

You are given an MxN structure array representing a zoo. Each row represents a section of the zoo (i.e. reptile house, primate pen, etc.), and each column represents a specific exhibit (i.e. salamander, komodo dragon, iguana, rattlesnake, etc.). Each exhibit has the following fields that contain the subsequent information: **`'exhibit'`**: *(char)* the name of the animal in the exhibit, '**`people'`**: *(double)* the average number of people who visit the exhibit per hour*,* **`'time'`**: *(double)* the average time, in hours, each person spends in the exhibit, **`'rating'`**: *(double)* the average rating (out of 5 stars) visitors gave the exhibit, and **`'ugaRank'`**: *(double)* a number representing the old ranking of the exhibit.

Your first job is to update each exhibit with a popularity score under a new field called **`'popularity'.`** You are given the helper function **`calcPopularity`**, which will do this for you. To find out how this function works, type **`help calcPopularity`** into the command window. You should also remove the ranking assigned to the exhibits by the u[sic]ga grad by removing the entire field **`'ugaRank'`**. Then remodel the zoo such that the most popular section is the first row, and the least popular section is last.

Continued…

Example:

For the following example, each box represents an index in the structure array and each line represents a different field: exhibit, people, time, rating, and rank or popularity respectively.

Given the input **`Zoo`**, your function should output **`remodeledZoo`**.

`Zoo`

`'Penguin'`
`10`
`.5`
`2`
`1`
	`'Polar Bear'`
	`10`
	`1`
	`2.5`
	`2`

`'Iguana'`
`12`
`.4`
`5`
`3`
	`'Salamander'`
	`12`
	`.4`
	`5`
	`4`

`'Tucan'`
`10`
`.1`
`1`
`5`
	`'Flamingo'`
	`17`
	`.75`
	`4`
	`6`

`remodeledZoo`

`'Tucan'`
`10`
`.1`
`1`
`1`
	`'Flamingo'`
	`17`
	`.75`
	`4`
	`51`

`'Iguana'`
`12`
`.4`
`5`
`24`
	`'Salamander'`
	`12`
	`.4`
	`5`
	`24`

`'Penguin'`
`10`
`.5`
`2`
`10`
	`'Polar Bear'`
	`10`
	`1`
	`2.5`
	`25`

Notes:

- The total popularity of a section can be found by summing the popularities of each individual exhibit.
- If two sections have the same overall popularity, then they stay in the same order as the original zoo

Hints:

- Structure arrays can be sorted similarly to other arrays, like cell arrays.

Function Name: `zooBreak`

Inputs:

1. (*struct*) An MxN structure array representing the city blocks
2. *(char)* A string specifying which animal has escaped the zoo

Outputs:

1. *(struct)* An MxN structure array representing the city blocks with the originally escaped animal in the zoo
2. *(double)* The Cartesian distance from the zoo to the escaped animal

Background:

It's finally the weekend and you want to relax, so you decide to go to Zoo Atlanta. While exploring the zoo, you realize that one of the animals is not in its exhibit. You are informed that the animal has escaped the zoo, and is hiding somewhere in the city. Your job is to find the missing animal and bring it back to the zoo.

Function Description:

You are given an MxN structure array, where each structure represents a city block. Each city block contains the fields **`'Place'`** and **`'Inhabitants'.`** You are also told which animal has escaped the zoo. The zoo may be anywhere in the city. You must look through every city block to find the missing animal, and you must add this animal to the zoo's inhabitants. You must also calculate the Cartesian distance from the city block containing the zoo to the city block containing the escaped animal. This distance can be calculated with

```
distance = sqrt( (block2X - block1X)^2 + (block2Y - block1Y)^2)
```

After finding the animal and adding it to the zoo, it should be removed from the city block it was found in by deleting it from the **`Inhabitants`** field.

Example:

In this example, the zoo is in city block (2,1), and the shark is in the city block (2,3). The Cartesian distance from the zoo to the shark is 2 city blocks.

```
city =
Place: 'park'                    Place: 'school'              Place: 'mall'
Inhabitants: 'birds'             Inhabitants: 'students'      Inhabitants: 'shoppers'

Place: zoo                       Place: 'library'             Place: 'river'
Inhabitants: {'snake', 'tiger'}  Inhabitants: 'books'         Inhabitants: 'shark'
```

Continued…

After finding the shark, it should be added to the end of the zoo's inhabitants list.

```
city(2,1) = Place: zoo
            Inhabitants: {'snake', 'tiger', 'shark'}
```

Notes:

- The escaped animal is guaranteed to be somewhere in the city, and will not already be in the zoo.
- Every structure will have at least one inhabitant. Only the zoo will have more than one inhabitant.
- The inhabitant list will be a character array for all structures except the zoo, which will be a cell array of character arrays.
- Do not round the distance calculation.

Hints:

- To calculate the Cartesian distance between city blocks, think about how the row and column of a structure in the array can relate to X and Y coordinates.

Function Name: `saveTheWhales`

Inputs:

1. (*struct*) A 1x1 structure containing information about the whales' population

Outputs:

1. *(struct*) A 1x1 structure updated with complete data

Background:

The Georgia Aquarium staff had a double-feature movie night for *Blackfish* and *Born to be Free* which triggered a major policy reversal stating that "the Georgia Aquarium will no longer take dolphins or whales caught in the wild". Instead, they've hired interns to track the population (in hundreds) of migrating whales over the course of a few summer weeks. Unfortunately, the aquarium hired u[sic]ga graduate students who made a total mess of the data. Now, you (a bright, young GT student!) must save the data and SAVE THE WHALES!

Function Description:

The fields within the given structure store the weekly population count given by a double, organized by letters (because u[sic]ga grads, eye roll), as shown below. You must determine which weeks are missing (weeks **`'D'`**, **`'F'`**, and **`'G'`** in the example below) and interpret what data should be there. Use the adjacent weeks to fill in evenly spaced data that would make sense for the missing weeks. From **`'C'`** week to **`'E'`** week, **`'D'`** would be 3.0 (directly between 2.5 and 3.5), and from **`'E'`** week to **`'H'`** week, **`'F'`** and **`'G'`** would be 3.7 and 3.9, respectively (evenly spaced between 3.5 and 4.1). Update the structure to include these new fields and data.

Example:

```
Structure = 'A': 1                    New Fields and Data
            'B': 2                          'D': 3
            'C': 2.5                        'F': 3.7
            'E': 3.5                        'G': 3.9
            'H': 4.1
```

Notes:

- Round population data to 2 decimal points.
- The first and last week of data will always be included'.
- Data will never go below zero (no negative populations).
- Fields don't have a particular order within the structures, so if your output structure isn't displayed in alphabetical order, that is ok.

Hints:

- You may find the **`linspace()`** function helpful in this problem.

Function Name: `predator`

Inputs:

1. (*struct*) A 1xM structure array of animals
2. (*cell*) An Nx1 cell array containing a list of catastrophes the animals have to survive

Outputs:

1. *(char*) A sentence describing the results of the observation

Background:

It's the year 2027! It's been ten years since you took CS 1371, and it has definitely prepared you well for your new job as the head dinosaur keeper for the Atlanta Zoo. (There are dinosaurs in the future.) Unfortunately, you don't remember much from the EAS class you took your first semester, so you'll probably have to brush up on your ecological knowledge. Luckily, the latest release of MATLAB, R2027b, has a handy **`timeTravel()`** function that allows you to travel 150 million years into the past for a prime opportunity to observe animals surviving in their natural habitats.

Function Description:

The function should take in a structure array of all the animals at the beginning of the observation period. Each animal will be represented by a structure with the following fields:

`Name`
`Species`
`Predator`
`SwimmingAbility`
`SurvivalSkill`
`Cuteness`
`Intimidation`
`Wisdom`

The **`Name`** and **`Species`** fields contain strings representing the animal's name and species, respectively. The **`Predator`** field contains the name of the species that is most dangerous to the animal. The remaining fields each contain a single double value representing the animal's strength in each area.

Each month, a disaster will occur. The second input will be an Nx1 cell array containing the event for each month. After each month, at least one animal will die, according to the rules in the following table.

Continued...

Event	Value in Cell Array	Description
Meteor Strike	**`'Meteor Strike'`**	The animal with the lowest value of **`Intimidation`** minus **`Cuteness`** will not survive.
Flood	**`'Flood'`**	The animal with the lowest sum of **`SwimmingAbility`** and **`SurvivalSkill`** will not survive.
Famine	**`'Famine'`**	All animals with predators that are still alive will die due to the food shortage.
Volcanic Eruption	**`'Volcanic Eruption'`**	All the animals with less than (not inclusive) the average **`SurvivalSkill`** will die.

For the events that require finding the minimum of some score (i.e. Meteor Strike and Flood), only one animal will die. If multiple animals share the same minimum, the animal with the lowest sum of all scores will be the one who dies.

If multiple animals survive, output the sentence:

`'<Number of animals remaining> animals survived <number of months> grueling months.'`

If only one animal survives, output the sentence:

`'After <number of months> months of observation, only <last animal's name>, a <last animal's species>, survived.'`

If no animals survive, output the sentence:

`'<Number of months> months were too much for the animals, and none of them survived.'`

Notes:

- No two animals will have the same sum of scores.
- Events might repeat.
- Each animal will have only one predator.