Function Name: `fluxCapacitor`

Inputs:

1. (*double*) The current year
2. (*double*) Destination year
3. (*double*) Speed, in miles per hour

Outputs:

1. *(char*) String stating whether you're going back or forward into the future

Background:

Doc Brown has invited you on an adventure through time. As you two hop into the time-travelling Delorean, he inputs the current year and the destination year. As the Doc steps on the gas, he reminds you that you can't interact with ANYONE, or you'll risk changing the flow of time itself! Midway through the speech, he realizes that the car might not reach the necessary speed to travel through time, so you'll need to quickly determine if you'll make it through time.

Function Description:

Write a function that takes in the current year, a destination year, and the max speed that the car can reach on the runway. Using these values, determine whether you'll be going back in time, to the future, or straight into a building if the speed is less than 88 mph.

- If you're going forward in time, output:
  `'We're going to the year <Destination Year>, to the future!!!'`
- If you're going back in time, output:
  `'We're going to the year <Destination Year>, back in time!!!'`
- If you're going under 88 mph, you won't be able to time travel, so output:
  `'Time traveling is just too dangerous. Better that I devote myself to study the other great mystery of the universe: MATLAB!'`

Example:

```
out = fluxCapacitor(2008, 3000, 89);

>>out => 'We're going to the year 3000, to the future!!!'
```

Notes:

- The current year and destination year will never be the same.
- To create a character vector with an apostrophe, use two apostrophes in a row.

Hints:

- Use the values from the first movie for a sweet surprise!

Function Name: `sandlot`

Inputs:

1. (*double*) The distance the ball was hit
2. (*double*) Angle in which the ball was hit, in degrees
3. (*char*) A string describing what pitch was thrown

Outputs:

1. (*char*) A string describing the outcome of your at bat

Background:

"You're killin' me, Smalls." Ham's words echo around in your head. You know you aren't the best baseball player on the team, but you can use MATLAB to help improve your game.

Function Description:

Use the angle of the hit to determine where the ball goes. The angle is measured with 0 degrees pointing at third base and 90 degrees pointing at first base. The field is divided into thirds as shown below:

Angle	Field sector	Distance to fence
$0 \le \theta \le 30$	`'left'`	`106 yards`
$30 < \theta < 60$	`'center'`	`136 yards`
$60 \le \theta \le 90$	`'right'`	`104 yards`

If the pitch was a `'knuckleball'` or a `'slider'`, you miss the ball and get a strike. In this case, output `'Strike!'`.

If the ball was hit a distance greater than or equal to the distance to the fence for the field sector that the ball is in, your output string should simply be `'Home run!'`. If the ball is not hit beyond the fence, your output string should have the following format:

```
'You hit the ball <distance hit> yards into <sector> field.'
```

Additionally, consider the case when θ < 0 degrees or θ > 90 degrees, corresponding to a foul ball. In this case, output the string `'Foul ball.'`

Example:

```
str = sandlot(100, 70, 'fastball')
str => 'You hit the ball 100 yards into right field.'
```

Notes:

- Makes sure to consider the order the conditionals are evaluated.

Function Name: `jurassicPark`

Inputs:

1. (*char*) Your move
2. (*char)* Your sibling's move

Outputs:

1. *(char*) A statement describing who won the game

Background:

Your parents have finally allowed you and your sibling to go visit your grandpa's new, revolutionary attraction… Jurassic Park, the surreal world full of dinosaurs. Once you get there, your sibling immediately wants to go see the Velociraptors! However, you, as a seasoned dinosaur expert, know that the much bigger and cooler, albeit less well-known, relative of the Velociraptor, the Utahraptor needs to be the first stop on your trip. Unable to convince each other, you two resort to a classic game of "T-rex, Velociraptor, Stegosaurus, Pterodactyl, Brachiosaurus"- it's like Rock, Paper, Scissors, but with dinosaurs!

Function Description:

Write a MATLAB function that will determine the winner of each round of T-rex, Velociraptor, Stegosaurus, Pterodactyl, Brachiosaurus by comparing the two strings given as inputs. These are the ONLY inputs we will use to test your function:

```
'T-rex'
'Velociraptor'
'Stegosaurus'
'Pterodactyl'
'Brachiosaurus'
```

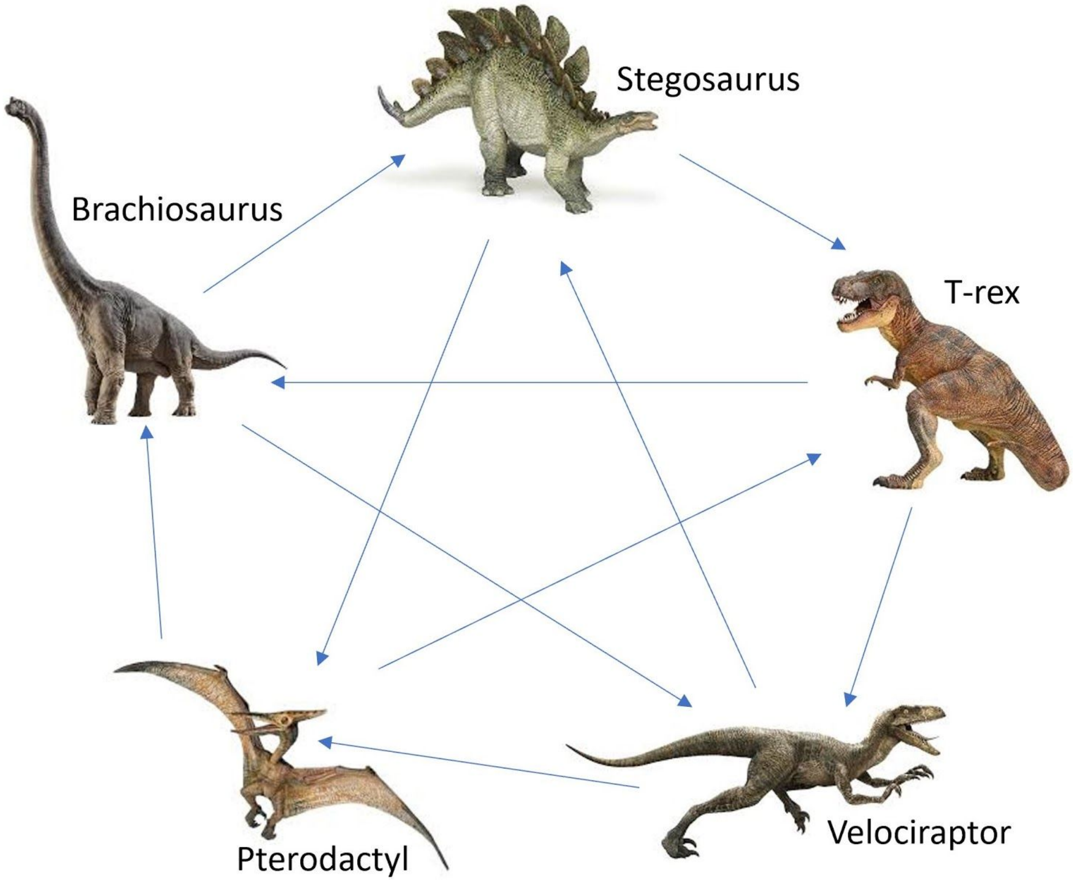
Then compare them by the following rules:

1. `'T-rex'` beats both `'Velociraptor'` and `'Brachiosaurus'.`
2. `'Velociraptor'` beats both `'Pterodactyl'` and `'Stegosaurus'`.
3. `'Stegosaurus'` beats both `'T-rex'` and `'Pterodactyl'`.
4. `'Pterodactyl'` beats both `'Brachiosaurus'` and `'T-rex'.`
5. `'Brachiosaurus'` beats both `'Velociraptor'` and `'Stegosaurus'`.

Continued...

After determining who wins, output one of the following strings:

- If you win:
    - `'<Your move> beats <Your sibling’s move>! Let’s go see the Utahraptors!'`
- If your sibling wins:
    - `'<Your move> loses to <Your sibling’s move>. I guess the Utahraptors will have to wait.'`
- If you both choose the same move:
    - `'It’s a tie!'`



Example:

```
>> winner = jurassicPark('Velociraptor', 'Stegosaurus')

winner =
      'Velociraptor beats Stegosaurus! Let’s go see the Utahraptors!'
```

Function Name: `spaceJam2`

Inputs:

1. (*double*) 1 x 2 vector of the max age and min height in inches, in that order
2. (*char*) Data from potential players application

Outputs:

1. (*char*) A string indicating whether the player made the team.

Background:

After a humiliating defeat against Bugs Bunny and his friends, the aliens are back for a rematch. The aliens have been training 24/7 for this epic rematch, while the Looney Tunes have been spending all their time studying for their MATLAB course. With Michael Jordan retired for good this time, the Looney Tunes must recruit some serious basketball talent to be on their team, so they only take the best players who meet certain standards.

Function Description:

Write a function that takes in a player's application data as a string and outputs a string stating whether they made the team or not. A player's application data will be in a string containing the following characteristics in this order, with each response separated by commas:

```
'<Name>,<Age>,<Height>,<Previous Team>,<Sponsor>,<Average Points Per Game>'
```

First, set a maximum age and a minimum height standard, specified by the first input. Anyone who does not meet these standards will automatically not be considered. A player's height is given in feet and inches formatted as such `'<feet>-<inches>'` separated only by a single dash.

If the player meets these two conditions, then they must either average at least 15 points per game, or be sponsored by Warner Bros.

If, however, the name on the player's application is `‘Lola Bunny’` or `'LeBron'`, they automatically make the team.

After analyzing all these criteria, you will output a simple string: `'<Name> made the team!'` OR `'Sorry, <Name> doesn’t have what it takes to space jam.'`

Example:

```
>> Data = ‘Patrick Star,46,5-2,Super Stars,Nickelodeon,17’
>> result = spaceJam2([70,60],Data)
      result => ‘Patrick Star made the team!’
```

Notes:

- Warner Bros may be referred to as `'WB'`, `'Warner Bros'`, `'Warner Bros Pictures'`.
- The test cases given **are not** exhaustive. Be sure to test all possible conditions!
- Try putting `‘Michael Jordan’` in the name of the application string and run the solution.

Function Name: `oceans11`

Inputs:

1. *(char)* a vector describing your hand
2. *(char)* a vector describing the dealer's hand

Outputs:

1. *(char)* a vector describing the game

Background:

You are Danny Ocean. Instead of robbing a bunch of casinos, you decide to use your best friend MATLAB to see if you can win mad ca$h at blackjack. To get you started on your path to fortune, you write code to determine who the winner of a round is.

Function Description:

You are given two vectors of space separated numbers and letters representing a set of cards. (e.g. `'Q 10 4'`) As in normal blackjack, all face cards (J, Q, K) have the value 10, and Aces (A) count as either 11 or 1. Your inputs will be either two or three cards long. The output of your function will be three sentences. The first sentence will describe your points.

Number of points	Statement
> 21	`'I busted.'`
21	`'I got a Blackjack!'`
< 21	`'I got a score of <score>.'`

The second sentence will describe the dealer's points.

Number of points	Statement
> 21	`'The Dealer busted.'`
21	`'The Dealer got a Blackjack!'`
< 21	`'The Dealer got a score of <score>.'`

Continued...

The third and final sentence will describe who won the round. Keep in mind that the dealer wins ties.

Case	Statement
I won	`'I win!'`
Dealer won	`'The Dealer wins!'`
Both Busted	`'We both busted.'`

Example:

```
>>oceans11('8 9', 'Q A')

↳ 'I got a score of 17. The Dealer got a Blackjack. The Dealer wins!'
```

Notes:

- For simplicity, you will never have more than 1 Ace in either hand.
- Counting an Ace as 11 is always more beneficial, unless the 11 causes you to bust.
- This is a good opportunity for you to explore creating a helper function to get the score from the input vector

Hints:

- One possible way to account for the face cards (J, Q, K) would be to use the function `strrep()` to replace the letters with their value before computing the value of the hand.
- `str2num()` can take in a space separated character vector of numbers and create a vector of doubles (e.g. `'3 10 11'` => `[3 10 11]`)