Function Name: `alphaIndex`

Inputs:

1. (*char*) A single character

Outputs:

1. (*double*) The numerical index of the input character within the alphabet

Background:

You heard in lecture that characters in MATLAB are actually stored as numeric values according to the ASCII Table. Let's put that knowledge to use!

Function Description:

Write a function that converts a character to its index in the alphabet (e.g. A = 1, B = 2, C = 3, …, Z = 26).

Examples:

```
ind1 = alphaIndex('e')
ind1 → 5

ind2 = alphaIndex('M')
ind2 → 13
```

Notes:

- The input character may be uppercase or lowercase.

Hints:

- What function(s) could help you guarantee which case you need to work with?

Function Name: `quadForm`

Inputs:

1. (*double*) The value of a
2. *(double)* The value of b
3. *(double)* The value of c

Outputs:

1. (*double*) The larger output of the quadratic formula
2. *(double)* The smaller output of the quadratic formula

Background:

You keep having to solve the quadratic formula over and over again in your math class when you realize - MATLAB could do it for you!

Function Description:

Given constants a, b, and c, compute the two x values that are the solutions to the quadratic formula.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Use the **plus** root for the first output and the **minus** root for the second output.

Example:

```
[x1, x2] = quadForm(5, -3, -2)
x1 → 1
x2 → -0.4
```

Notes:

- Round both outputs of your function to 2 decimal places.
- You are guaranteed that all cases used to test your code will produce real solutions.

Function Name: `EOS`

Inputs:

1. (*double*) Volume (L)
2. (*double*) Number of moles
3. (*double*) Temperature (K)

Outputs:

1. (*double*) Maximum pressure (atm)
2. (*logical*) If ideal gas pressure is larger than van der Waals pressure or not

Background:

All your life you've used the ideal gas law to calculate gas properties, until your professor tells you there are so many more equations of state that are much more accurate! Interested in investigating the validity of the ideal gas law, you decide to compare it to the van der Waals equation of state.

Function Description:

Given volume (V), number of moles (n), and temperature (T) of a sample of hydrogen, find the pressure (P) of the gas using the ideal gas law and van der Waals equation of state.

Ideal gas law:

$$PV = nRT$$

Van der Waals equation of state:

$$\left(P + \frac{a}{V_m^2}\right)(V_m - b) = RT$$

with molar volume V_m defined as:

$$V_m = \frac{n}{V}$$

Use R = 0.0821 as the gas constant, and a = 0.242 and b = 0.02651 for hydrogen.

Output the greater of the two pressures, rounded to the nearest whole number, as well as a logical stating whether the ideal gas law pressure is greater than the van der Waals pressure or not (`true` if IG > VDW, and `false` if IG <= VDW).

Example:

```
[maxP, log] = EOS(5, 2, 300)
maxP → 64
log → false
```

Hints:

- The `max()` function will be helpful!

Function Name: `pizzaParty`

Inputs:

1. *(double)* Number of party attendees
2. *(double)* Number of pizzas ordered

Outputs:

1. *(double)* Slices of pizza per person
2. *(double)* Slices of pizza left over

Background:

You decide to host a party and buy some pizzas for you and your friends. But, in order to be fair, every person should get the same number of slices of pizza. Any extra slices will be left over and saved for later.

You could try and calculate the pizza slice distribution yourself, but why bother when you can have MATLAB do all the hard work!

Function Description:

Given the number of people at the party and the number of pizzas ordered, write a function that outputs how many slices of pizza each person should receive, and how many slices of pizza will be left over, assuming every pizza is cut into 8 pieces.

Example:

```
[num, left] = pizzaParty(5, 2)
num → 3
left → 1
```

Notes:

- Assume every pizza is cut into 8 slices.
- You cannot have fractional slices.

Hints:

- You may find the `floor()` and/or `mod()` functions useful.

Extra Credit

Function Name: `citizenWatch`

Inputs:

1. *(double)* The current position of the hour hand
2. *(double)* The current position of the minute hand
3. *(double)* A positive or negative number of minutes

Outputs:

1. *(double)* The position of the hour hand after the specified time
2. *(double)* The position of the minute hand after the specified time

Background:

The moment is finally here. You've been saving up your whole life and now you can finally afford a Citizen watch! You decide to go to Ross to buy your very own, brand new Citizen watch. Unfortunately, your friend Sam presses a random button on your Citizen watch and you're worried that he broke it. To make sure he didn't, you decide to write a MATLAB function to determine where the hour and minute hands would be after a certain amount of time has passed so that you can make sure your Citizen watch is still working properly.

Function Description:

This function will take in the current position of the hour hand, as an integer between `0` and `11` (`0` for noon/midnight), the current position of the minute hand, as an integer between `0` and `59` (`0` for "on-the-hour") and a positive or negative number of minutes elapsed.

Given this information, determine the new position of the clock hands. You should assume that the hour hand does not move until the next hour has begun. For example, the hour hand stays on `2` from 2:00 until 2:59 and only at 3:00 does the hour hand move to `3`.

Example

```
[hour, min] = citizenWatch(11, 30, 122)
hour → 1
min → 32
```

Notes:

- The `mod()` and `floor()` functions will be useful.
- As you do this problem, notice the behavior of `mod()` when the first input is negative.

Hints:

- Information on using mod in this manner:
  - Khan Academy - Better Explained