Drill Problem #1

Function Name: gravCalc

Inputs:

1. (*double*) The mass of an object (in kilograms)
2. (*double*) The distance from the center of the Earth to the center of the object (in meters)

Outputs:

1. (*double*) The force of gravity between earth and the object

Function Description:

Write a function that calculates the force of gravity between the Earth and some other object. The formula for gravity is:

$$F_g = G * \frac{m_1 * m_2}{d^2}$$

`m1` will always be the mass of the earth, which is 5.97e24 kg. `m2` will be the input mass of the other object. `d` is the input distance. You can assume this will be the distance between the centers of mass of the two objects. `G` is the universal gravitation constant, which is 6.67e-11 $\frac{m^3}{kg \cdot s^2}$. You can also assume all inputs will be in SI units and your output should be in Newtons, $1\ N = 1\ \frac{kg \cdot m}{s^2}$ (no unit conversions will be necessary).

Notes:

- You will want to use 5.97e24 instead of 5.97x10^{24}. If you do not use e, your answer will be off because of how MATLAB approximates calculations.

Drill Problem #2

Function Name: distCalc

Inputs:

1. (*double*) The x-coordinate of a given point A
2. (*double*) The y-coordinate of point A
3. (*double*) The x-coordinate of another point B
4. (*double*) The y-coordinate of point B

Outputs:

1. (*double*) The distance between points A and B

Function Description:

Write a function, `distCalc`, in MATLAB, that will calculate the distance between the two Cartesian points defined in the inputs. The formula for calculating distance between 2 points is:

$$dist = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Round your answer to the hundredths place.

Drill Problem #3

Function Name: lawOfCosines

Inputs:

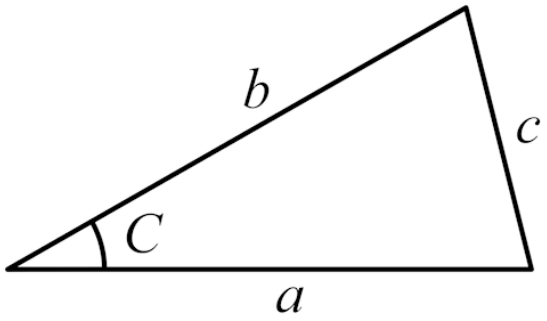
1. (*double*) The length of side *a*
2. (*double*) The length of side *b*
3. (*double*) The angle between sides a and b (C in the diagram), in degrees

Outputs:

1. (*double*) The length of side *c*

Function Description:

Write a MATLAB function that will solve for the length of the third side of a triangle (*c in the picture)* given the opposite angle and two other side lengths, as shown in the picture below.



Notes:

- The law of cosines (as the name of the problem implies) may prove to be very helpful.
- For reference:

$$c^2 = a^2 + b^2 - 2abcos(C)$$

- MATLAB has a built in function for the evaluation of cosine in degrees: `cosd()`

Drill Problem #4

Function Name: tippingPoint

Inputs:

1. (*double*) The subtotal cost of a meal before taxes or tips
2. (*double*) The sales tax for a meal at a given food establishment, as a decimal
3. (*double*) The percent being tipped to the waiter, as a percentage

Outputs:

1. (*double*) The total tip amount before tax
2. (*double*) The total tip amount after tax

Function Description:

You go to dinner with your friends, and in between eating and discussing quiz bowl, you realize that if you apply the tip for the waiter before the sales tax is applied, it will be different than if applied after the sales tax.

Using the sales tax and the tip percentage, write a MATLAB function, `tippingPoint`, to calculate:

- The total amount you would tip for your meal if you calculated the tip before applying sales tax.
- The total amount you would tip for your meal if you calculated the tip after applying sales tax.

Output these values, rounded to the hundredths (cents) place.

Drill Problem #5

Function Name: spherePacking

Inputs:

1. (double) The sphere radius
2. (double) The length of the side of a cube

Outputs:

1. (double) The number of spheres that will fit into the cube

Function Description:

The problem of sphere packing (how many spheres will fit into a given volume) is actually a non-trivial problem in mathematics and was only proven in the 1990's. If you would like to read more about the problem, you can follow these links:

http://mathworld.wolfram.com/SpherePacking.html
https://en.wikipedia.org/wiki/Sphere_packing

Otherwise, you are welcome to take for granted that the maximum packing density of spheres in a cube is 74.048%. That means that under the best packing scenario, the spheres will occupy 74.048% of the cube's volume. Given this and the radius of a sphere and the edge length of a cube, write a MATLAB function, `spherePacking`, to calculate the maximum number of spheres that will fit in the cube. You cannot have fractional spheres.

Notes:

- The `floor()` function may be useful.

Drill Problem #6

Function Name: letThemEatCake

Inputs:

1. (*double*) The number of total pieces of cake
2. (*double*) The number of people at your party

Outputs:

1. (*double*) The number of pieces of cake you will get

Function Description:

You are having a birthday party, and are trying to predict how many pieces of cake you'll get to eat by yourself. Of course, your first instinct is to write a MATLAB function to do this for you! There are two inputs: the total number of pieces of cake, and the number of people at the party. You know that the cake will need to be divided evenly between all guests at your party, including yourself, and that you want everyone to eat the same number of cake pieces. Because of this, there might be a few pieces of cake left over that could not be fairly distributed. These few pieces left over will be yours to eat after everyone has left, so you will get all of them. Output the total number of cake pieces you get to eat. The number of people at the party will always include yourself.

For example,if there are 20 pieces of cake and 6 people at your party: 18 pieces of cake will be distributed among the 6 people so that each person gets 3 (including yourself). This will leave 2 pieces of cake remaining from the original 20, and you will also get these 2 pieces. So your output, in this case, should be 5.

Notes:

- The `mod()` and `floor()` functions may be useful.