Function Name: `solveQuad`

Inputs:

1. *(double)* The coefficient of the ‘x^2’ term in a quadratic polynomial
2. *(double)* The coefficient of the ‘x’ term in the quadratic polynomial
3. *(double)* The constant term in the quadratic polynomial

Outputs:

1. *(double)* The first root of the quadratic polynomial
2. *(double)* The second root of the quadratic polynomial

Function Description:

Remember all those nights in Algebra you spent solving quadratic after quadratic? Now that you’ve gone through that grueling process, you will never have to solve another one by hand because you can make MATLAB do it for you! This function will take in the 3 coefficients of a quadratic polynomial and output the roots.

Just to refresh your memory, the formula for the roots is shown below. ‘a’ corresponds to the first input to the function, ‘b’ is the second input, and ‘c’ is the third.

$$first\ root = \frac{-b+\sqrt{b^2-4ac}}{2a}$$

$$second\ root = \frac{-b-\sqrt{b^2-4ac}}{2a}$$

Notes:

- You will not be given any test cases that produce complex roots.
- You should round your answers to four decimal places.

Function Name: `inAndOut`

Inputs:

1. *(double)* The radius of a sphere

Outputs:

1. *(double)* The volume of a cube circumscribed around the sphere
2. *(double)* The volume of a cube inscribed in the sphere

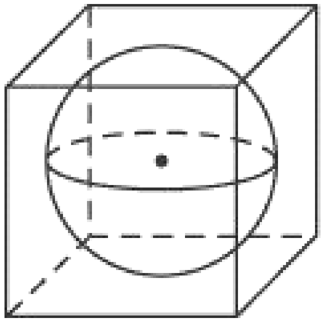
Function Description:

Write a MATLAB function that will take in a circle's radius and calculates the volume of two different cubes. The first cube is circumscribed around the sphere (the smallest cube that can completely enclose the sphere) and the second cube is inscribed in the sphere (the largest cube that can fit in the sphere). Don't worry if your geometry is a little rusty, because the formula for the edge length 'x' of a cube inscribed in a sphere of radius 'r' is:

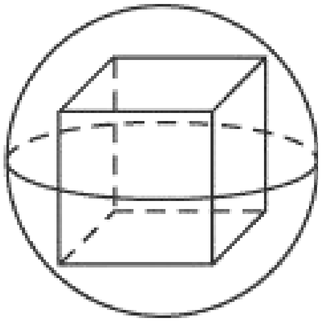
$$x = \sqrt{\frac{4r^2}{3}}$$

Here is a visual of the two cubes you will calculate the volume for:

Circumscribed cube



Inscribed cube



Round both of your answers to the nearest hundredth.

Function Name: `sepDigits`

Inputs:

1. *(double)* A three-digit integer

Outputs:

1. *(double)* The hundreds-place digit of the input
2. *(double)* The tens-place digit of the input
3. *(double)* The ones-place digit of the input

Function Description:

This function will take in a three-digit integer (integer between 100 and 999, inclusive) and return each digit as a separate number. For example, if the input is 472, the first output would be 4, the second would be 7 and the third would be 2.

Notes:

- You only have to deal with three-digit positive integers.
- The `mod()` and `floor()` functions will do all the heavy lifting.

Function Name: `clockHands`

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

Function Description:

It is not always immediately obvious where the hands of a clock will be after a certain amount of time. It is even harder to visualize where the hands *were* some amount of time in the past. Luckily, this is not a very difficult problem for a computer to solve, so you will use that to your advantage. This function will take in the current position of the hour hand, as an integer between 0 and 11, inclusive (0 for noon/midnight) and the current position of the minute hand, as an integer between 0 and 59, inclusive (0 for “on-the-hour”) and a positive or negative number of minutes. Given this information, calculate 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”.

Notes:

- The `mod()` and `floor()` functions will be useful.
- As you do this problem notice the behavior of `mod()` for negative inputs. This is a very important function in programming and will come up again in the class!

Hints:

- One way of solving this problem involves calculating the total number of minutes after noon/midnight before and after the given minutes have elapsed.
- Another way of solving it involves splitting the given number of minutes into a number of hours and a number of minutes.
- Pick whichever method makes more sense to you (or come up with your own method).