Function Name: `fallTime`

Inputs:

1. (*double)* Height of roof, in meters

Outputs:

1. *(double*) Time it takes the object to fall, in seconds

Background:

Standing on the roof of the CULC, you accidentally drop your cell phone over the edge! Before your phone reaches the ground, you decide to make a function in MATLAB to help you determine how long it will take to reach the ground.

Function Description:

An object under constant acceleration with initial velocity v_i (m/s) and acceleration a (m/s^2) traveling for a time t (s) travels a distance d (m) according to the following kinematics equation:

$$d = v_i t + \frac{1}{2}at^2$$

Given the height of the CULC, calculate t, the time it takes for your phone to reach the ground.

Notes:

- Since you drop (and don't throw) your phone, you can assume initial velocity, v_i, is 0 m/s.
- The phone is in free fall, so a is the acceleration due to gravity. a = g = 9.8 m/s^2.
- Round your answer to two decimal places.

Function Name: `cartDist`

Inputs:

1. *(double)* The first point's x-coordinate (x_1)
2. *(double)* The first point's y-coordinate (y_1)
3. *(double)* The second point's x-coordinate (x_2)
4. *(double)* The second point's y-coordinate (y_2)

Outputs:

1. *(double)* The distance between the two points

Function Description:

This function will take in two points defined in 2 dimensional space and calculate the distance between them. The points will be represented using Cartesian coordinates in the form [x_1, y_1] and [x_2, y_2]. In case you are rusty on your geometry, the distance between two points can be calculated using the following formula:

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

Notes:

- Round the distance to the nearest hundredth.

Function Name: `timeDilation`

Inputs:

1. (*double*) time measured by third party (in seconds)
2. (*double*) velocity (in meters/second)

Outputs:

2. *(double*) time measured by observer (in seconds)

Background:

Have you ever thought about how if you throw a ball on a moving train, the ball has a different speed relative to you compared to a stationary observer on the train platform as the train passes by? Well, if you hadn't already, now you have! Now it's *time* to make MATLAB help us with this calculation.

Function Description:

Here is the formula:

$$T = \frac{T_0}{\sqrt{1 - (\frac{v}{c})^2}}$$

where T is the time measured by the moving observer (s), T_0 is the time measured by the stationary observer (s), v is the velocity of the moving observer (m/s), and c is the speed of light.

Your function should take in the velocity, v, and the time measured by third party, T_0, and output the time measured by the observer, T.

Notes:

- This problem ignores time dilation as caused by relative differences in gravitational fields.
- For the speed of light, use $c = 299,792,458$ m/s.
- Round your answer to the fourth decimal place.

Function Name: `spaceTime`

Inputs:

1. (*double*) Number of Earth years
2. (*double*) Semi-major axis of the other planet's orbit (in m)
3. (*double*) Mass of the other planet (in kg)

Outputs:

1. (*double*) Number of years on the other planet

Background:

While watching the movie *The Martian*, you begin to wonder about how years on other planets relate to years on Earth. You find yourself curious about how time would pass differently on Mars, and on other planets. Since a year on Earth is how long it takes the Earth to complete an orbit around the sun, how would this compare to a year on Mars, or how long it would take Mars to make a complete orbit around the sun? Since you're an engineer, you decide to generalize this idea to work for any number of Earth years and any other planet, given readily available orbital and physical data.

Function Description:

Newton's law of gravitation gives the orbital period T of a body of mass M_1 around another body of mass M_2 with semi-major axis of orbit a as

$$T = 2\pi\sqrt{\frac{a^3}{G\,(M_1 + M_2)}}$$

where $G = 6.674 \times 10^{-11}$ m^3 kg^{-1} s^{-2} (the gravitational constant), a is in meters, T is in seconds, and M_1 and M_2 are both in kilograms.

Notes:

- A year is the orbital period of a planet, or the time it takes for the planet to make a complete orbit around the Sun.
- The semi-major axis of a planet's (elliptical) orbit is half the longest diameter of its orbit.
- The Sun's mass is 1.989×10^{30} kg.
- All planets given for this function will be within the Solar System, so none will be more massive than the Sun.
- Round the output to three decimal places.

Hints:

- All these worlds are ours, but there is one on which we should attempt no landing… try using the parameters of that world as the inputs to the solution function.

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

Background:

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.

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`.

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 and 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).