Computer Science 010: Design and Implementation of Solutions to Computational Problems

Episteme 9

This is an assignment should be done in pairs with the person assigned to you on Canvas. You may choose to do either the first or the second assignment. You don't have to do both. If

Program #1: Basic (max possible 95%)

Assignment:

Implement a class `Car` that can be used to simulate a car. It should have the following properties:

A car has a certain fuel efficiency (measured in miles/gallon) and a certain amount of fuel in the gas tank. The efficiency is specified in the constructor, and the initial fuel level is 0. Supply a method `drive` that simulates driving the car for a certain distance, reducing the fuel level in the gas tank, and methods `getGasLevel`, to return the current fuel level, and `addGas`, to add gas to the tank.

If driving the car results in running out of gas, then your method should raise a `ValueError` exception with the appropriate message.

Sample test code 1:

```
from Car import Car
myHybrid = Car(50)    # 50 miles per gallon
myHybrid.addGas(20)    # Tank 20 gallons
myHybrid.drive(100)    # Drive 100 miles
print(myHybrid.getGasLevel())    # Print fuel remaining -> 18
```

Sample test code 2:

```
from Car import Car
myHybrid = Car(50)    # 50 miles per gallon
myHybrid.addGas(1)    # Tank 1 gallons
myHybrid.drive(100)    # Drive 100 miles
<Exception "You ran out of gas">
```

You will turn in a file E_09_01.py which contains **just the class definition** I will test by running a program like the samples above

Program #2: Advanced (max possible 110%)

Assignment:

Implement a class `ComboLock` that works like the combination lock in a gym locker, as shown here:

The lock is constructed with a combination—three numbers between 0 and 39.

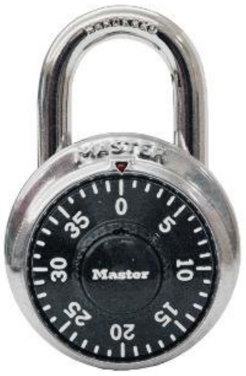
The `reset` method resets the dial so that it points to 0.

The `turnLeft` and `turnRight` methods turn the dial by *a given number of ticks* to the left or right.

The `open` method attempts to open the lock.

The lock opens if the user first turned it right to the first number in the combination, then left to the second, and then right to the third.

Here is a stub that you should complete:



```
class ComboLock :
   def __init__(self, secret1, secret2, secret3) :
      . . .
   def reset(self) :
      . . .
   def turnLeft(self, ticks) :
      . . .
   def turnRight(self, ticks) :
      . . .

   #Returns True or False
   def open(self) :
      . . .
```

You will turn in a file E_09_02.py which contains **just the class definition** I will test by running a program that uses it like this test code below:

```
from ComboLock import ComboLock
myLock = ComboLock(10,20,10)
myLock.reset()
myLock.turnRight(30)
myLock.turnLeft(10)
myLock.turnRight(10)
if(myLock.open() == True):
     print("The lock opened")  #Correct outcome
else:
      print("The lock didn't open")

myLock.reset()
myLock.turnLeft(30)
myLock.turnRight(10)
myLock.turnLeft(10)
if(myLock.open() == True):
     print("The lock opened")
else:
      print("The lock didn't open") #Correct outcome
```