Lab Exam

August 23, 2023

Q1

```
[ ]: def can_sell_tickets(money_list):
         ticket_price = 250
         change_available = 0

         for money in money_list:
             if money == ticket_price:
                 change_available += money
             elif money > ticket_price:
                 change = money - ticket_price
                 if change_available >= change:
                     change_available -= change
                 else:
                     return False
             else:
                 return False

         return True

     # Test cases
     test_cases = [
         [250, 250, 500],
         [250, 500, 500],
         [250, 1000],
         [250, 250, 500, 500]
     ]

     for idx, test_case in enumerate(test_cases):
         result = can_sell_tickets(test_case)
         print(f"Test case {idx + 1}: {result}")
```

q2

```
[ ]: class Vehicle:
         def __init__(self, brand, model, type, fuel_tank_size):
             self.brand = brand
             self.model = model
```

```
        self.type = type
        self.fuel_tank_size = fuel_tank_size
        self.fuel_level = 0

    def fulltank(self):
        self.fuel_level = self.fuel_tank_size
        print("Tank is Full")

    def update_fuel_tank(self, new_level):
        self.fuel_level = new_level
        if self.fuel_level <= 3:
            print("Warning: Low fuel level!")

    def get_fuel(self, amount):
        if self.fuel_level + amount <= self.fuel_tank_size:
            self.fuel_level += amount
        else:
            print("Warning: Fuel tank capacity exceeded!")

    def drive(self):
        print(f"WOW! I am Driving {self.model}")

# Create two vehicle objects
car1 = Vehicle("Toyota", "Camry", "Sedan", 60)
car2 = Vehicle("Honda", "Civic", "Sedan", 50)

# Call methods on the objects
car1.fulltank()
car2.update_fuel_tank(10)
car1.get_fuel(20)
car2.drive()

# Inheritance and Polymorphism
class ElectricVehicle(Vehicle):
    def __init__(self, brand, model, type, battery_capacity):
        super().__init__(brand, model, type, 0)  # Electric vehicles don't have␣
 ↪fuel tanks
        self.battery_capacity = battery_capacity

    def charge(self):
        print(f"Charging {self.brand} {self.model}")

    def drive(self):
        print(f"Zooming Quietly in {self.brand} {self.model}")

# Create an electric vehicle object
electric_car = ElectricVehicle("Tesla", "Model S", "Electric Sedan", 100)
```

```python

# Call methods on the electric vehicle object
electric_car.charge()
electric_car.drive()
```