Item 1

Questions | Data structures

You are tasked with developing a simple parking lot management system. The parking lot uses a camera to record the license plates of cars as they enter and exit, and these license plates are stored in two Python lists. The system should help manage tasks like tracking which cars are currently in the lot, checking if a specific car is present, and handling entries and exits.

For the following exercises, consider the lists `cars_in` and `cars_out`, which store the license plates of the cars that have entered and exit the parking lot throughout the day, respectively. The license plates are stored in ascending order of arrival/departure (i.e. the first car to arrive/exit is the first element of the list). For example:

```
cars_in = ["23-AB-45", "67-CD-89", "12-EF-34", "90-GH-12"]
cars_out = ["12-EF-34", "90-GH-12"]
```

Given these lists, answer the following questions.

Which of the following answers the question "What was the second car to enter the parking lot?"

A
```
cars_in[2]
```

B
```
cars_in[1]
```

C
```
cars_in[3]
```

D
```
cars_in[-2]
```

Which of the following answers the question "What is the license plate of the last car to exit the parking lot?"

A
```
cars_out[-2]
```

B
```
cars_out[0]
```

C
```
cars_out[-1]
```

D
```
cars_out[2]
```

Which of the following answers the question "How many cars have exited the parking lot?"

A
```
len(cars_in) - len(cars_out)
```

B
```
len(cars_in)
```

C
```
len(cars_out) - len(cars_in)
```

D
```
len(cars_out)
```

Which of the following answers the question "Is there any car in the parking lot"?

A
```
len(cars_in) == len(cars_out)
```

B
```
len(cars_in) - len(cars_out) == 1
```

C
```
len(cars_in) - len(cars_out) > 1
```

D
```
len(cars_in) - len(cars_out) > 0
```

Which of the following answers the question "Has the car 67-CD-89 entered the parking lot?":

A
```
"67-CD-89" not in cars_in
```

B
```
"67-CD-89" in cars_in
```

C
```
"67-CD-89" in cars_in or "67-CD-89" in cars_out
```

D
```
"67-CD-89" in cars_out
```

Which of the following correctly answers the question "Is the car with license plate 12-EF-34 currently in the parking lot?"

A
```
"12-EF-34" not in cars_out
```

B
```
"12-EF-34" in cars_in and "12-EF-34" not in cars_out
```

C
```
"12-EF-34" in cars_in or "12-EF-34" not in cars_out
```

D
```
"12-EF-34" in cars_in
```

Which of the following correctly records car 67-CD-89 exiting the parking lot?:

A
```
cars_in.remove("67-CD-89")
```

B
```
cars_out.add("67-CD-89")
```

C
```
cars_out.append("67-CD-89")
```

D
```
cars_out + "67-CD-89"
```

Item 2

Questions | Data structures

Suppose you now store the pollution levels of the cars that enter the parking lot in a dictionary. The possible pollution levels are "eco", "standard", and "toxic":

For the following exercises, consider the dictionary:

```
cars_emissions = {
    "23-AB-45": "eco",
    "67-CD-89": "eco",
    "12-EF-34": "standard",
    "90-GH-12": "toxic"
}
```

Which of the following answers the question "What is the pollution level of car 67-CD-89?"

A
```
cars_emissions["67-CD-89"]
```

B
```
cars_emissions("67-CD-89")
```

C
```
cars_emissions[1]
```

D
```
cars_emissions.get(1)
```

Which of the following answers the question "Is the pollution level of car 23-AB-45 toxic?"

A cars_emissions[0] = "toxic"

B cars_emissions[0] == "toxic"

C cars_emissions["23-AB-45"] = "toxic"

D cars_emissions["23-AB-45"] == "toxic" ✔

What is the output of the following code?

```
if "01-HV-12" in cars_emissions:
  print("Found")
else:
  print("Not found")
if "23-AB-45" in cars_emissions:
  print("Found")
elif "67-CD-89" in cars_emissions:
  print("Found")
if "71-TG-34" in cars_emissions:
  print("Found")
else:
  print("Not found")
```

A

```
Not found
Found
Not found
```

B

```
Not found
Found
```

C

```
Not found
Found
Found
Not found
```

D

```
Not found
Found
Found
Found
```

What is the output of the following code?

```
for car, level in cars_emissions.items():
  if level == "ECO" and level == "toxic":
    print(car)
```

A

```
23-AB-45

67-CD-89
```

B

```
23-AB-45

67-CD-89

90-GH-12
```

C

```
90-GH-12
```

D

Nothing ✔

What is the output of the following code?

```
i = 0
keys = list(cars_emissions.keys())
while i < len(keys):
  print(cars_emissions[keys[i]])
```

A

```
eco
eco
standard
toxic
```

B

```
23-AB-45
67-CD-89
12-EF-34
90-GH-12
```

C

Nothing

D

It enters an infinite loop printing

```
eco
eco
eco
...
```

Which of the following answers the question "How many eco cars have entered the parking lot?"

A

```
count = 0
  while count < len(cars_emissions):
    if cars_emissions[count] == "eco":
        count += 1
print(count)
```

B

```
count = 0
for emissions in cars_emissions:
    if emissions == "eco":
        count += 1
print(count)
```

C

```
count = 0
for emissions in cars_emissions.keys():
    if emissions == "eco":
        count += 1
print(count)
```

D

```
count = 0
for emissions in cars_emissions.values():
    if emissions == "eco":
        count += 1
print(count)
```

Which of the following correctly answers the question "Of the cars currently in the parking lot, which was the first to enter"? Remember that cars_in and cars_out are sorted in ascending order of arrival time.

A

```
for car in cars_in:
    if car not in cars_out:
        print(car)
        break
```

B

```
for car in cars_out:
    if car not in cars_in:
       print(car)
       break
```

C

```
print(cars_in[0])
```

D

```
print(cars_out[0])
```

Fill in the blanks to answer the question "How many standard cars have left the parking lot?"

```
count = 0
for car in [1]:
  if [2] == "standard":
    [3]
print(count)
```

Correct answers:

1 cars_out 2 cars_emissions[car] 3 count += 1

Fill in the blanks to answer the question "How many eco cars are **still** in the parking lot?":

```
count = 0
for car in [1]:
    if [2] [3] [4]:
        count += 1
print(count)
```

Correct answers:

1 cars_in 2 cars_emissions[car] == "eco" 3 and 4 car not in cars_out

Fill in the blanks to create a dictionary where the keys are the pollution levels and the values are the number of cars with that pollution level, for instance:

```
level_cars = {"eco": 2,
              "standard": 1,
              "toxic": 1}
```

```
level_cars = {"eco": 0,
              "standard": 0,
              "toxic": 0}
for car in [1]:
  pollution_level = [2]
  [3][ [4] ] += 1
```

Correct answers:

1 cars_emissions 2 cars_emissions[car] 3 level_cars 4 pollution_level

Item 3

Questions | Data structures

Given the following function for counting how many cars of a given pollution level are in the parking lot:

```
def count_cars_by_pollution_level(level="eco"):
    ...
```

Which of the following options **throws an error** when calling the function?

A
```
count_cars_by_pollution_level(level="toxic")
```

B
```
count_cars_by_pollution_level("standard")
```

C
```
count_cars_by_pollution_level()
```

D
```
count_cars_by_pollution_level(pollution_level="eco")
```

Suppose the parking lot applies a tax on the base price for cars that are not eco. The base price is 15EUR/hour and the penalties are:

- 30% for standard cars
- 100% for toxic cars

Fill in the blanks to complete the definition of a function named `hourly_price` that takes one argument, `license_plate`, and computers the hourly price for the car according to its pollution level and the penalties.

```
def hourly_price( 1 ):
  price = 15
  pollution_level = 2
  if 3 :
    return price * 1.3
  elif 4 :
    return price * 2
  else
    return price
```

Correct answers:

1 license_plate 2 cars_emissions[license_plate] 3 pollution_level == "standard"

4 pollution_level == "toxic"

The parking lot also applies a **10% discount for stays over 8 hours**.

Fill in the blanks to complete the definition of a function named `ticket_price` that takes two arguments, `license_plate` and `time` (the number of hours the car was parked), and returns the total ticket price for the car, based on the hourly rate and discount applied (if any). Use the previous function, `hourly_price`, to aid in the computation:

```
def ticket_price( 1 ):
  price = hourly_price( 2 ) * time
  if 3 :
    return price * 0.9
  else:
    return price
```

Correct answers:

1 license_plate, time 2 license_plate 3 time > 8

Which of the following correctly defines the function `profit` that takes one argument, `hours`, a dictionary where keys are license plates and values are the number of hours they stayed in the parking lot, and computes the profit of the parking lot. This value is obtained by adding the ticket price, for the length of their stay, of all the cars that have exited the parking lot.

```
def profit(hours):
  for car in cars_out:
    return ticket_price(car, hours)
```

```
def profit(hours):
  total = 0
  for car in cars_out:
    total += ticket_price(car, hours)
    return total
```

```
def profit(hours):
  total = 0
  for car in cars_out:
    total += ticket_price(car, hours[car])
  return total
```

✔

```
def profit(hours):
  total = 0
  for car in cars_out:
    total += ticket_price(car, hours)
  return total
```

Considering the following function

```
def plates(all=False):
  for car in cars_out:
    if not all:
      print(car)
    else:
      print(car)
      return car
```

What is the output of the following code?

```
plates(True)
```

A

```
12-EF-34
90-GH-12
```

B

```
12-EF-34
12-EF-34
90-GH-12
90-GH-12
```

C

```
12-EF-34
```

D

```
90-GH-12
```

Item 4

Questions | Data structures

Consider the following class to represent a car with its license plate and pollution level:

```
class Car:
  def __init__(self, license_plate, pollution_level):
    self.license_plate = license_plate
    self.pollution_level = pollution_level
```

Implement the `ParkingLot` class to represent a parking lot as described earlier.

The class attributes are:

- `capacity` : An integer with the maximum number of cars allowed at the same time in the parking lot
- `cars_in` : A list with the license plates of cars that have entered the parking lot
- `cars_out` : A list with the license plates of cars that have exited the parking lot

Fill in the blanks to implement the class constructor. The constructor takes one argument, capacity, used to initialize the capacity attribute. The attributes cars_in and cars_out are initialized as empty lists.

```
class ParkingLot:
  def __init__( 1 ):
      2
      3
      4
```

Correct answers:

1 self, capacity 2 self.capacity = capacity 3 self.cars_in = [] 4 self.cars_out = []

Which of the following answers the question "Create a new parking lot with capacity for 150 cars"?

A park = ParkingLot(self, 150)

B park = ParkingLot(capacity=150) ✔

C park = ParkingLot(capacity=150, cars_in=[], cars_out=[])

D park = ParkingLot()

Fill in the blanks to complete the definition of method `taken_spots` in the `ParkingLot` class, to compute the percentage (e.g. 50%) of taken spots in the parking lot at a given moment, depending on its capacity and the number of cars inside. This method takes no arguments.

```
class ParkingLot:
  ...
  def taken_spots( 1 ):
    cars_inside = 2
    return cars_inside / 3 * 100
```

Correct answers:

1 self 2 len(self.cars_in)-len(self.cars_out) 3 self.capacity

Which of the following correctly defines the method `register_car_enter` that takes one argument, `car` (an object of the `Car` class), and adds its license plate to the `cars_in` attribute?

A

```
def register_car_enter(car):
    cars_in.append(car)
```

B

```
def register_car_enter(self, license_plate):
  self.cars_in.append(license_plate)
```

C

```
def register_car_enter(self, car):
  self.cars_in.append(car.license_plate)
```

D

```
def register_car_enter(self, car):
  cars_in.append(car.license_plate)
```

What is the output of the following code?

```
park = ParkingLot(20)
car  = Car("19-OM-21", "eco")
park.register_car_enter(car)
print(park.cars_in)
```

A Error

B `[]`

C `["19-OM-21"]`

D Nothing