Data Structure and Object Orientated Development

Question One

Lists

Lists can be used to manage collections of related objects. In object-oriented programming (OOP), lists are often used to represent a group of objects. For example, a Library class might use a list to store instances of Book.

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

class Library:
    def __init__(self):
        self.books = []

    def add_book(self, book):
        self.books.append(book)
```

Dictionaries

Dictionaries facilitate the mapping of keys to values, which is useful in OOP when attributes or behaviours need to be accessed using a unique identifier. This might be used to store details about employee against a ID.

```
self.employees = {
    101: {"name": "Alice", "position": "Manager", "department": "HR"},
    102: {"name": "Bob", "position": "Developer", "department": "IT"},
    103: {"name": "Charlie", "position": "Analyst", "department": "Finance"}
}
```

Stack (LIFO – Last in First out)

Stacks support reverse sequential processing, which is crucial for tasks like backtracking or managing function calls. In OOP, a stack could be implemented to handle undo/redo operations or the execution order of recursive algorithms.

Question Two

Create a nested dictionary of data on cars within a Car class. Extend the program to work with the dictionary by calling the following methods:

- items()
- keys()
- values()

```python
class Car:
    def __init__(self):
        self.car_data = {
            "sedan": {"brand": "Toyota", "model": "Camry", "year": 2020},
            "SUV": {"brand": "Ford", "model": "Explorer", "year": 2021},
            "truck": {"brand": "Chevrolet", "model": "Silverado", "year": 2022},
        }

    def display_items(self):
        return self.car_data.items()

    def display_keys(self):
        return self.car_data.keys()

    def display_values(self):
        return self.car_data.values()

# Example usage
car = Car()
print("Items:", car.display_items())
print("Keys:", car.display_keys())
print("Values:", car.display_values())
```

```
Items: dict_items([('sedan', {'brand': 'Toyota', 'model': 'Camry', 'year': 2020}), ('SUV', {'brand': 'Ford', 'model': 'Explorer', 'year': 2021}), ('truck', {'brand': 'Chevrolet', 'model': 'Silverado', 'year': 2022})])

Keys: dict_keys(['sedan', 'SUV', 'truck'])

Values: dict_values([{'brand': 'Toyota', 'model': 'Camry', 'year': 2020}, {'brand': 'Ford', 'model': 'Explorer', 'year': 2021}, {'brand': 'Chevrolet', 'model': 'Silverado', 'year': 2022}])
```