CS2106 - Data Structure and Algorithms 17.12.2024 Time:60minutes

Question 1: Game Event Processing (Queue)
Scenario:
You are managing a game server that processes **player actions**. Actions arrive in a sequential order, and the system must process them in the same order (FIFO).
Task:

Implement a **queue** data structure to simulate action processing.Define the following operations:

- enqueue(action): Adds a player action to the queue.
- dequeue(): Processes the oldest action in the queue and removes it.

```
enqueue("Player 1 shoots")
enqueue("Player 2 jumps")
dequeue()   // Should process: "Player 1 shoots"
dequeue()   // Should process: "Player 2 jumps"
```

Question 2: Dungeon Escape (Stack)
Scenario:
In a dungeon crawler game, players need to keep track of rooms they've visited. Each room leads to other rooms, and players can use a **Stack** to backtrack to previous rooms if they hit a dead end.
Task:

1. Use a **stack** to implement the player's movement between rooms.
2. The following operations should be supported:
    - visitRoom(roomName): Push a room to the stack (mark as visited).
    - backtrack(): Pop the current room to go back to the previous one.

```
visitRoom("Entrance")
visitRoom("Hallway")
visitRoom("Treasure Room")
backtrack()  // Back to "Hallway"
backtrack()  // Back to "Entrance"
```

Question 5: Battle Queue (Queue)

Scenario:
In a multiplayer game, players are waiting in a queue to battle in a tournament. The server processes them in a **First Come, First Serve** manner.
Task:

Implement the tournament queue using a **queue data structure**.
Define the following operations:

`joinQueue(playerName)`: Add a player to the queue.
`startBattle()`: Start a battle with the first player in the queue and remove them.

```
joinQueue("Player_A")
joinQueue("Player_B")
startBattle()  // "Player_A" battles
startBattle()  // "Player_B" battles
```