Quiz 02:

Duration: 40 minutes
Total Problems: 2

Problem 1: Queue-Based Ride Allocation System (20 minutes)

You're designing a simple ride allocation system for an amusement park. The rides operate in fixed rounds, with each round accommodating a set number of people. Visitors wait in line, and the queue is managed to ensure that everyone is served efficiently.

  - Implement a circular queue where each "Enqueue" represents a visitor joining the line and each "Dequeue" represents a visitor being seated for a ride.
  - The ride has a capacity limit (given as input). Once the capacity is reached in a single round, dequeue the respective number of visitors.
  - Any extra visitors wait for the next round.
  - After each ride round, the queue rotates once to simulate the passing of time, shifting everyone forward.

- **Requirements**:

  1. Prompt the user for the number of visitors joining the queue.
  2. Output the state of the queue after each round, indicating who's left in line and who's seated.
  3. Manage full and empty queue conditions.

- **Example**:
  - Input: Queue size = 5, Round capacity = 2
  - Operations: Enqueue Visitor1, Visitor2, Visitor3, Visitor4, Visitor5
  - Output:

Round 1: Seated: Visitor1, Visitor2 | Remaining Queue: Visitor3, Visitor4, Visitor5
Round 2: Seated: Visitor3, Visitor4 | Remaining Queue: Visitor5
Round 3: Seated: Visitor5 | Remaining Queue: (Queue is now empty)

Problem 2: Stack-Based Directory Traversal and Path Simplification (20 minutes)

You're building a file system utility that helps simplify and validate a given path in a directory. The path may include:

- Directory names (e.g., `docs`, `images`)
- `".."` (indicating moving up one level in the directory hierarchy)
- `"."` (indicating the current directory, which should be ignored)

Your task is to use a stack to handle these components and simplify the path to an absolute form. If the path starts with `/`, consider it as the root directory. Assume directory names only contain alphanumeric characters.

- **Requirements**:
  1. Write a function that simplifies paths like `"/home/../user/docs/./photos"` to `"/user/docs/photos"`.
  2. Handle cases where `".."` appears at the root directory (e.g., `"../../file.txt"` should return `"/file.txt"`).
  3. Use the stack to push directory names, pop when encountering `".."`, and ignore `"."`.
- **Input Examples**:
  - Input: *"/projects/../user/./docs/notes"*
    - Expected Output: *"/user/docs/notes"*
  - Input: *"../../work/files/./"*
    - Expected Output: *"/work/files"*