Master “C Language” in 30 Days Challenge

(By Tech Involvers)

Project 4: Pac Man Game

Instructions:

- Read the problem carefully before trying to solve it.
- Do the tasks on your own. Don’t copy it.
- The output of your program must be the same as given in the sample run.

<u>Overview</u>:

Pac-Man is a classic arcade game where the player navigates Pac-Man through a maze, eating dots while avoiding ghosts. The objective is to eat all the dots in the maze while avoiding being caught by the ghosts.

<u>How to Implement</u>

1. Game Loop

- The game runs in a continuous loop, updating the game state and rendering the game screen in each iteration.
- The loop will handle user input, update the positions of Pac-Man and the ghosts, check for collisions, and render the game state on the screen.

2. Game State Management

- The game state includes the positions of Pac-Man, ghosts, dots, and power-ups, as well as the player's score and lives.

3. Rendering

- Use a graphics library to render the game. Libraries like SDL (Simple DirectMedia Layer) or Pygame can be used for rendering graphics and handling user input.

4. User Input

- Capture keyboard input to control Pac-Man's movement (up, down, left, right).

5. Collision Detection

- Check for collisions between Pac-Man and walls, dots, ghosts, and power-ups.
- Handle collisions appropriately, such as stopping movement when hitting a wall, eating a dot, or losing a life when caught by a ghost.

6. Game Over and Winning Conditions

- The game ends when Pac-Man loses all lives or when all dots are eaten.
- Display appropriate messages and handle restarting or quitting the game.

<u>Requirements</u>

Master “C Language” in 30 Days Challenge

1. **Maze Layout**
    - Design a maze layout with walls, dots, and power-ups.
    - Store the layout in a 2D array or a similar data structure.
2. **Pac-Man**
    - Implement Pac-Man's movement controlled by user input.
    - Ensure smooth movement and collision detection with walls.
3. **Dots and Power-Ups**
    - Place dots and power-ups in the maze.
    - Implement logic for Pac-Man to eat dots and power-ups.
4. **Ghosts**
    - Ghosts should move around the maze, either randomly or following a specific algorithm to chase Pac-Man.
5. **Scoring System**
    - Track the player's score based on the number of dots eaten.
    - Implement bonus points for eating power-ups and ghosts.
6. **Lives and Game Over**
    - Track the number of lives Pac-Man has.
    - Implement game over conditions and display a game over screen.
7. **User Interface**
    - Display the current score and number of lives on the screen.
    - Provide a start screen and game over screen.