**Introduction.**

This report provides a detailed explanation of the problem, the solution approach, and the code used to determine the winner of the labyrinth race among three wizards.

**Understanding the Labyrinth.**

The labyrinth is a grid where:

# - represents walls.

. - represents open paths.

E - represents the exit.

**Problem Description**

The competition involves three wizards trying to find the exit of a labyrinth as quickly as possible. Each wizard starts at a different position and has a unique speed. The goal is to predict which wizard will reach the exit first, assuming they all follow the shortest path to the exit.

**Solution Approach**

To solve the problem, we perform a single Breadth-First Search (BFS) from the exit position to calculate the shortest distance from the exit to every other cell in the labyrinth. This approach allows us to determine the shortest path distances efficiently. We then use these distances to calculate the time each wizard takes to reach the exit based on their speed. The wizard with the shortest time is declared the winner.

**Explanation of BFS**

**Reverse BFS** *reverse\_bfs:* This function starts from the exit and calculates the shortest distance from the exit to all other cells in the labyrinth. It uses a queue to process each cell and updates distances for its neighboring cells if they are within bounds, not a wall, and haven't been visited yet. **Prediction** *predict\_winner*:The exit position is identified. The reverse\_bfs function is called once to get the distances from the exit to all cells. For each wizard, the distance from their starting position to the exit is retrieved from the precomputed distances, and the time to exit is calculated based on their speed. The wizards are sorted by their time to exit, and the winner is determined. Using this method, BFS is executed exactly once, making the solution efficient while ensuring accurate results.

**Code Explanation**

The solution involves the following steps: Find the exit position: traverse the labyrinth to locate the exit cell marked with 'E'. Perform reverse BFS from the exit: calculate the shortest distances from the exit to all cells using BFS. Calculate times for each wizard: using the precomputed distances, determine the time each wizard takes to reach the exit based on their speed. Determine the winner: identify the wizard with the shortest time to reach the exit.

**Conclusion**

This solution efficiently determines the fastest wizard using BFS for shortest path computation and straightforward calculations based on provided speeds. It handles typical edge cases and ensures accurate prediction of the winner in labyrinth escape scenarios.