***secretary:*** *Sara* ***Host:*** *Anitta* ***Scribe:*** *Karim*

---

## ***Session 4:*** **Collaborative Development and Performance Evaluation of Maze Generation and Resolution Algorithm**

***introduction:*** **I**n this session, we gathered all the code and tests we had previously developed, combining them into a cohesive solution. Each team member presented the bugs or limitations they had identified during individual testing, which allowed us to make targeted improvements to the code. We refined the implementation to ensure it was complete, accurate, and aligned with all project guidelines.

## ***Code modifications and improvement:***

- ***Enhanced Maze Display:*** One of our main updates was enhancing the maze's visual representation in the terminal to clearly demonstrate the pathfinding process. The initial maze display was modified to show the distinct paths covered by each BFS:
- Path from the door (`door`): marked with `*`
- Path from the exit (`exit`): marked with `#`

This distinction allowed for better visualization of the bidirectional search and the eventual convergence point of the two BFS processes, highlighting the shortest path from the `door` to the `exit`.

- ***Edge Case Handling:*** Our code now accounts for several edge cases to ensure robustness:
- Invalid Maze Sizes: Checks were added to reject maze dimensions that are impractical or non-generable, such as `(1, 0)` or `(0, 1)`, `(0, 0)` which would be impossible to construct.
- Door Positioned at Exit: A special case was implemented to handle scenarios where the starting point (`door`) coincides with the exit (`exit`). In such cases, the algorithm directly concludes the solution without performing a BFS, as the shortest path is trivial (already at the exit).

- ***Exploration of Display Alternatives:*** We explored using SDL to improve maze visualization, allowing for clearer rendering of paths, doors, and exits with interactive graphics compared to static output.

## ***Performance Evaluation and Testing:***

**1. Test for Algorithm Efficiency on Different Maze Sizes:**
Evaluate the algorithm's execution time and memory usage for varying maze sizes →karim

**2. Test for Pathfinding Accuracy:**
Ensure the algorithm finds the correct shortest path from the door to the exit. sizes →lucas

**3. Test for Bidirectional Search Convergence:**
Check if the bidirectional search correctly converges from both the door and exit. →Daniel

**4. Test for Maze Generation with Random Walls:**
Verify that random maze generation produces solvable and valid mazes.
→anitta

**5. Test for Edge Case Handling:**
Test how the algorithm handles edge cases, like when the door is at the exit.
→Tinting

**6. Memory Usage Test:**
Monitor memory consumption during maze generation and pathfinding for large mazes to ensure efficiency. →Sara