Individual Assignment Report

Student name: Dominik Duc Duy Nguyen

1. Logic of BotDomDeep

My bot navigates each map systematically, moving from top to bottom and left to right, with the following priority logic:

- **Immediate Stain Cleaning:** If a stain is detected above or below the bot, it will clean the stain immediately and return to its original position.
- **Row Traversal:** When no stains are in sight, the bot moves right on even-numbered rows and left on odd-numbered rows. Upon reaching either the far left or far right column (wall), the bot skips ahead by two rows. For example, after completing the first row, the bot jumps directly to the fourth row.
- **Obstacle Handling (Wall Mode):** If an obstacle is detected within its vision, the bot activates "wall mode," suspending the row-skipping behavior. In this mode, the bot maneuvers around obstacles by moving downward or reversing direction

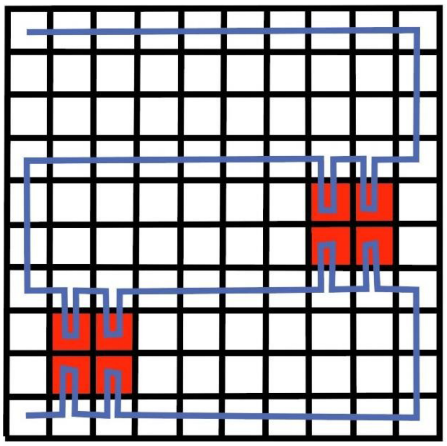

Figure 1: Regular movement example of my bot with row skipping. Red squares are stains, black squares are obstacles.

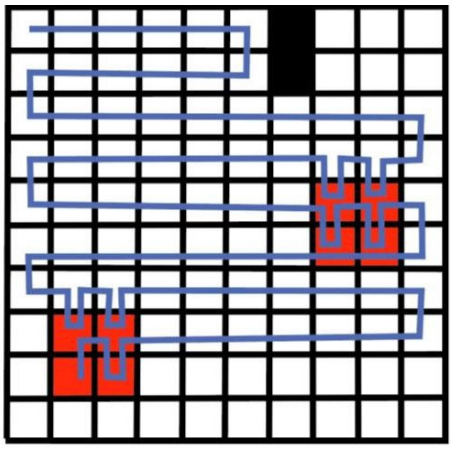

Figure 2: "wall_mode" movement example of my bot without row skipping. Red squares are stains, black squares are obstacles

2. Performance

I tested my bot on all the maps provided in the graded_maps.zip file. It successfully solved the 6-grade and 7-grade maps, as well as all 8-grade maps except one. The bot also managed to complete one 9-grade map. However, the remaining 9-grade maps and the Labyrinth (10-grade) maps presented challenges. The bot got stuck at "oddly positioned" horizontal pillars and was unable to backtrack the cells it had skipped or missed due to obstacles. In some cases, overlapping commands caused the bot to get stuck as we

3. Discussion and potential improvements

Before achieving the final results, I explored several different approaches:

- **Surrounding Cell Cleaning**: Initially, I programmed the bot to clean all cells within its 3x3 vision (UP, DOWN, RIGHT, LEFT). However, this often led to scenarios where the bot became stuck, unable to efficiently move forward.
- **Tracking Visited Cells**: Next, I implemented a system to track a set of visited_cells, allowing the bot to potentially backtrack if needed. Unfortunately, this did not yield successful outcomes, as the bot struggled to navigate effectively through the map.
- **Returning to Initial Cell**: I also attempted to have the bot return to its original position after cleaning stains, but this strategy proved unsuccessful and led to inefficient movements.
- *Improved Approach - A Search Algorithm**: A more robust solution would involve integrating an algorithm like *A search**, using heuristics to optimize pathfinding and allow the bot to prioritize efficient routes while avoiding obstacles.
- **Heat Map Strategy**: Another potential improvement could be creating a heatmap of the map's grid, assigning values to each cell based on its importance. This would guide the bot to prioritize high-value areas while avoiding unnecessary movements toward less critical zones.