Word Count: 655

Individual Report – CS1006 Project 3: Nonogram

Overview & Contributions:
During the Nonogram project, I was heavily involved in both the design and implementation of the game model, and in building several of the secondary and tertiary requirements. My main responsibilities revolved around implementing the core model for the Nonogram puzzle, handling undo/save/load features, and developing the full logic behind the solver and guesser.

I worked closely with my teammates through consistent peer programming sessions, especially when tackling challenging components such as the solver. As a team, we communicated primarily over WhatsApp and managed our repository using Git on the University server. Most of my work was committed directly with detailed messages explaining logic changes and architectural decisions.

What I Did:

1. Model of the Nonogram:
I developed the main model classes that represented the puzzle structure. This included how constraints were stored and interpreted, and how the grid maintained FILLED, EMPTY, and UNKNOWN cell states. I ensured that the model allowed clear interactions between the UI, checker, and solver components. The data structures I created allowed for efficient updates and checks, which became especially important for implementing undo and for the solver's recursive calls.

2. Undo, Save, and Load (Secondary Requirements):
A significant portion of my time was dedicated to move tracking and file I/O for recording and applying player actions.

- I implemented a move stack that recorded each action (cell update) by the player.
- I created JSON serialisation methods to save these moves into a separate file, adhering to a clean and readable format which was similar to the format in which we were provided the files.
- I also created a move loader which applied the saved move sequence to a loaded puzzle. It included handling exceptions gracefully if the file was malformed or incompatible with the puzzle.
  This feature was tested thoroughly through repeated undo chains and save/load cycles.

3. Tertiary Requirements – Solver and Guesser:
Implementing the solver was the most intellectually challenging part of my contribution. It was implemented with peer programming; it was challenging to critically think and implement it. We had written the logic with pen and paper multiple times, and it made complete sense but implementing the logic was really hard to think about the most effective and efficient solution.

- My partner and I started by building a deductive solver using recursive depth-first search, which attempted to fill in lines that had only one valid configuration based on the current known values.

- Once we had that working, I extended it into a full guesser with backtracking. When deduction failed, it would make a guess, proceed recursively, and revert if the guess led to inconsistency.
- This part required significant debugging and testing across edge cases like multi_checks.json and player.json. We verified the correctness of the solver using internal logging and custom JSON puzzles designed to test branching and backtracking.

Team Dynamics & Peer Programming:
We used pair programming extensively, especially during implementation of complex features like the the solver. At times, I was the driver, writing the recursive logic and testing it, while my partner navigated by reading the spec and helping detect flaws in the logic flow. At other times, we reversed roles, and I got to review and suggest improvements in their implementation. This iterative approach helped improve both speed and accuracy and made debugging easier. I personally learned a lot from this process.

Final Thoughts:
Overall, I feel satisfied with my contributions. I handled some of the most challenging and rewarding parts of the project, from designing the game model to implementing the solver and undo system. I got the chance to improve my skills in recursion, Java file I/O, and collaborative version control. While there were bugs and setbacks during development, especially in integrating UI and logic, we managed to fix most of them through consistent communication and division of responsibilities. If I had more time, I would have tried to optimise the solver further for performance.