Report of Asg1 (Tetris)

My code encompasses several notable features, including row elimination, HTML child clearing, seed, exceeding the top, behavior subject, rotation, replay, hard drop, hold, and a slight addition of wall kick functionality, totaling 6 points special implementation and 4 points additional features. Organized across six files—main, observable, types, view, state, and util—the logical functions are stored in the state and util files, where util houses commonly used functions. All required features perform as expected, with some variation which players earn points: one row clears for 100 points, two rows for 300 points, three rows for 500 points, and four rows for 800 points, all multiplied by the current level. Leveling up occurs after every 10 rows cleared. The state updates are managed by two functions, reduceState and scan. Each observable passes an action and the original state to these functions, and the collected data is then sent to the rendering process. After game end, player able to play next game by pressing next and also there is replay button just next to it to replay the last game player played, in replay mode player able to exit and play new game any time by pressing the new game at bottom-left which under the replay mode text. Pressing C for hold, Space for hard drop. Down for accelerate downward speed.

In my implementation, there are several key aspects to highlight. Firstly, I handle row elimination in a unique way by storing all stopped cubes in a single array, making this part somewhat complex. I create an array with predefined y positions for each row, like [0, 20, 40...380], which map to the cubes in those rows later. I then check the row lengths, filter out the full rows, and take additional actions to update the old stop array. Finally, I flatten them into an array that needs to be removed during rendering. Secondly, for child clearing, instead of using the conventional HTMLElement.removeChild() method, I employ HTMLElement.innerHTML = '' to clear the canvas, preview, and hold elements. Thirdly, I avoid using Math.random() in my code by using new Date().getTime() to obtain a long-type timestamp as the seed. Fourthly, considering the instruction to start dropping blocks from the top, I assume that blocks cannot render outside the canvas. I handle exceeding the top by detecting collisions. Since I already manage collisions in the handleVerticalCollision function, any remaining collision indicates that the block spawns with overlap with other blocks. In such cases, the block is automatically moved up by one cube's distance until it no longer exceeds the top. This approach improves rendering but has a tradeoff, as the I-shaped block cannot move to the very top of the canvas due to instant collision with the block beneath it upon spawning. Fifthly, I implement the use of a BehaviorSubject, which I taught myself, to handle complex updates such as difficult updating (switching to a new interval) and restarting the game by switching to a new game stream. Sixthly, Rotation are simulating SRS system but I calculate the position after rotation using the algorithm of Vector rotation in Mathematics.

Lastly, there are four additional features, one of which is not completely implemented. Firstly, the partially implemented feature is the wall kick, which currently only includes right and left wall kicks. When a rotation occurs next to a wall, the block shifts one cube's distance to the left or right. Secondly, I've incorporated a game replay function that allows players to review their last game. To achieve this, I've used BehaviorSubject and ReplaySubject. BehaviorSubject stores each game, but players can only replay their most recent one, while ReplaySubject records every state during the game, albeit with a tradeoff where states replay at a constant speed due to the replay is through zipping with an interval. Thirdly, I've included a hard drop feature, enabling users to press the spacebar to instantly move a block down to the bottom until it reaches the end or collides. Lastly, there's the hold feature, allowing players to temporarily store the current moving block by pressing 'C' and retrieve it for moving block by pressing 'C' again. These last two features are standard features from the Tetris game.