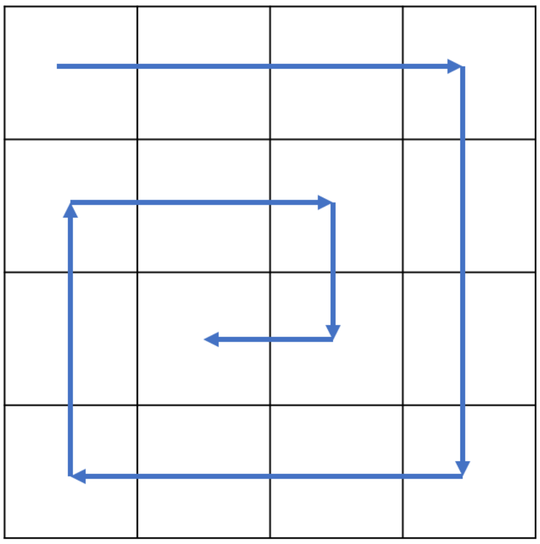
Programming Tasks (Extension)

Extension 1

The current helix pattern sequence is a representation of a 3 × 3 section of the grid. This limits the range of letters which can be represented with the space available for symbols. Introduce new functionality to allow the application to use a 4 × 4 section of the grid to represent a pattern sequence. Use this sequence to draw the helix pattern:



Extension 2

The application currently is a single-player puzzle. Introduce new functionality to allow the puzzle to be played by two players. A two-player puzzle should only be playable in a standard puzzle (8 × 8 grid). Players should take it in turns to place symbols into the grid. Although both players can use the same patterns and symbols (from a single SymbolsLeft total), player 1 places upper-case symbols and player 2 places lower-case symbols. Players gain points by placing either a complete, valid upper-case pattern (for player 1) or a complete, valid lower-case pattern (for player 2). A pattern which contains both upper- and lower-case symbols is not valid. The game ends when the symbols have been exhausted. The winner is the player with the highest score.

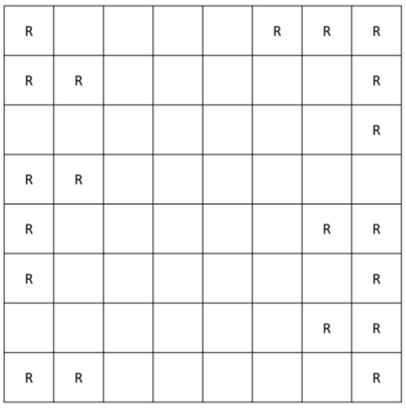
Extension 3

Currently the user must decide where they would like to place a symbol to create patterns. Introduce new functionality for the computer to 'suggest a move'. This costs the user 5 points. Before each turn, offer the user the option to 'view a move suggestion'; the computer should calculate a possible option for where a completed pattern (any of the symbols) can be placed into the grid. This should be shown to the user (option here for making the suggested option a different colour). After viewing the move suggestion, the console is cleared, the user score is reduced by 5 and the grid is returned to its original state, allowing the user to then place symbols into the grid.

Extension 4

The implementation of the pre-release material uses a square grid both with the external puzzle files and the standard puzzle. Introduce functionality into the standard puzzle to have the option of ‘rock walls’ on the left and right of the grid. Create a new class called WallCell which is placed at random down the left and right of the grid when the grid is created. A WallCell operates in the same way as a BlockedCell. A user cannot move a WallCell or place a symbol onto it.

WallCells should be placed at a random depth of 0 to 3 cells deep from the left or right of the grid. Blocked cells should still be placed as normal into the empty cells remaining after the walls have been placed.



R					R	R	R
R	R						R
							R
R	R						
R						R	R
R							R
						R	R
R	R						R

Extension 5

In the current implementation of the pre-release, if the user enters an invalid name for an external file, the application fails to load the file, which creates an empty grid which they cannot interact with. Introduce new error handling which is tolerant of this fault. On entering an invalid name for an external file, the application should give the user a second chance to re-enter the filename by giving a suitable error message. If the user enters an invalid name for a second time, the application should alert the user to this and instead load a standard puzzle.

Extension 6

Building a pattern by placing symbols one at a time can take a long time. Create a new method in the Puzzle class called PlaceWholePattern. On each turn, give the user the option to ‘place a whole pattern’. When selected, the application should ask for the row and column location of the top-left cell of the pattern helix and pass these to the PlaceWholePattern method, which should place the whole pattern into the grid. As per the current rules of the application, the method should not place symbols into invalid locations. Once a valid pattern has been placed, the PlaceWholePattern method should check for a pattern match and update the score accordingly.

Extension 7

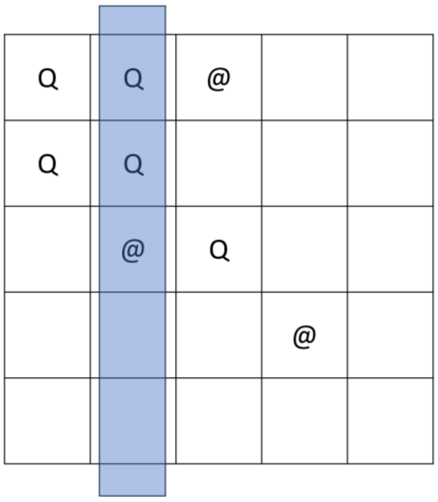
Introduce a new type of cell called a ‘Pattern Glitch’ which appears at random (50% chance of showing) each turn. The ‘Pattern Glitch’ places itself in a random cell on the board. If that cell is already part of a pattern, then the pattern is ‘glitched’, which means that the player loses the score for that pattern. The player can re-overwrite a ‘Pattern Glitch’ with a symbol to regain their score, but uses up a symbol in doing so.

Extension 8

In the current implementation of the pre-release application, the puzzle is finished when the user runs out of symbols to be placed into the grid. Introduce new functionality which counts all the empty cells at the end of the puzzle and deducts that from the score. If a cell is within the pattern area of two patterns, it is worth two negative points. This encourages the player to have more interlocking patterns to reduce the number of overlapping empty cells.

Extension 9

Introduce a new type of cell called ‘Line Bonus’. Only one ‘Line Bonus’ can appear on the grid per puzzle, positioned randomly. The ‘Line Bonus’ is awarded after the user places the first valid pattern. A ‘Line Bonus’ can be on either a single row or a single column. The example shown is shown on column 2.



When a ‘Line Bonus’ is awarded, advise the user which column or row it has been placed in. Any symbols which are on that line and in a pattern get an extra 5 points. Any symbols on that line which are not part of a pattern don’t gain any extra points. At the end of the puzzle, the ‘Line Bonus’ is added to the player score.

Extension 10

In the current implementation of the pre-release application, the puzzle UI is completely black and white because exam centres may not have access to colour printers. Introduce colour into the puzzle to highlight partial and complete patterns. Change the blocked cell symbol to red. When a symbol is placed into the grid it should be orange. When a pattern is matched, all the symbols in that matched pattern should be changed to green.

Extension 11

The current implementation of the pre-release application uses either a standard blank puzzle or four external puzzle files which can be loaded into the application. Create a new method in the Puzzle class called PuzzleBuilder which allows the user to manually build a new puzzle grid cell by cell. The method should ask the user what size puzzle they would like to build with a valid GridSize range between 3 and 9. The method should then allow the user to build a grid from the top left to the bottom right, asking the user what they would like to place in each cell. The method must comply with the rules for the number of blocked cells placed within the grid. A user can place any symbols into the grid, allowing them to build a partially complete grid. On completion of the grid, the method should calculate the score for that grid if the user has placed any matching patterns. The method should then ask the user how many symbols the puzzle should start with to be available to place into the grid and then give the option to save their built puzzle grid.

Extension 12

In the current implementation of the pre-release application, all matched patterns have the same value. Introduce new functionality so that different patterns are given different scores. The score for each pattern should be chosen at random between 10 and 15 points when the application starts, and the user should be advised. When a user places a valid matching pattern, the score should be updated appropriately and the points awarded for that pattern should be decremented by one. This means that placing a valid Q pattern into the grid may award 14 points for the first Q pattern. Placing a second valid Q pattern into the grid awards 13 points. Placing a third valid Q pattern into the grid awards 12 points and so on.