Use of data structures:

As required by the spec we Implemented a custom LinkedList and used it represent each player's hands, as well as the tilebag. The LinkedList allowed efficient addition to the back of the list and removal from the front of the list with popFront and pushBack, although access and removal from an arbitrary part of the list with removeAt becomes less efficient than a vector or an array when using a LinkedList. Because the majority of the actions performed on the player's hand are arbitrary removals, it would make more sense to implement it with a set data structure in the future. The tilebag on the other hand is a perfect fit for the LinkedList, as it only ever experiences removals from the front and additions at the back.

We used a vector to store the player objects for the game, which allowed an arbitrary number of players to join the game, as well as the ability to add them to the container one at a time and have the container resize dynamically on its own. The vector also allows convenient use of foreach loops (for (Player player : players)) due to its implementation of iterators, allowing us to cleanly perform the same operation for each player in the game.

We used a 2D array of tile pointers to represent the board, allowing random access to any position in the board efficiently. The array contained tile pointers rather than tiles to allow nullptr to represent an empty space.

In the validity checking of testPlacement, unordered sets were used to keep track of which tiles had already appeared when checking through a segment for duplicated tiles, more efficiently than if a vector had been used due to the constant time to check if an element is contained within the set vs. linear time for checking if an element is contained within a vector.

Finally, we used classes and structs to package together data for easier transport and manipulation as well as providing some basic functionality, with the Placement, Position, and Players classes.

Group coordination and project management:

In the project we used GitHub for source control, Trello for task management and Facebook/Messenger for communication, all three services functioning well in fulfilling their respective tasks. Week to week we came together during the tut, discussing the changes we'd made, any problems we'd had and distribute the work equally for the week ahead. For the GameEngine and LinkedList, work was split up by function, whereas the enhancements were split up as whole tasks, with Angelo and Mili each working on their own enhancement, and Flynn and Ye Wyn working on two enhancements together. Each piece of work was given a complexity rating out of 5, with the intention that functionality would be distributed such that at the end of the project each team member would have roughly the same total complexity rating, and hence had completely roughly the same amount of work. The final complexity scores are: Flynn = 14, Mili = 15, Angelo = 14 and Ye Wyn = 14, indicating a reasonably even distribution of work amongst the team. All pieces of functionality and associated complexity ratings can be found on the Trello page, but do not that all functionality for phases 0 through 2 were completed before the enhancements were implemented, and so any extra functions found in the code are due the enhancement development, and hence fall within their complexity ratings rather than generating their own.

Test case explanation:

placeTile1:

- Shows that attempting to place a tile in a position outside the board (place R5 at F26), a tile that does not exist in the players hand (place R6 at E1), or a tile on top of another tile (place O1 at C5) will all fail.
- Shows that attempting to place a tile next to one that shares no similarity (place R5 at L1) or next to a duplicate tile (place O1 at L1) will fail.
- Shows that placing a tile next to another that is either the same shape (place O1 at C6) or the same color (place R1 at G1) will succeed.

- Shows that placing a tile next to a segment will add to the score or each tile in the segment including the placed tile itself (O6 at M5), and that placing a tile next to two segments will add to the score for every tile in both segments, counting the placed tile twice (R6 at F5).
- Shows that if a qwirkle is achieved then QWIRKLE!!! is printed out and an extra 6 points is awarded

placeTile2:

- Shows that placing a tile next to a segment in which the adjacent tile is valid, but already contains the specified tile, will fail (Y2 at G1). Also shows that placing a tile next to a segment in which the adjacent tile is valid, but the segment's similarity type is different to the similarity between the specified tile and the adjacent tile, will fail (Y2 at H2).
- Shows that placing a tile between two segments that share a similar tile is invalid (R4 at B3) but placing a tile at the corner of two segments that share a similar tile is fine (R4 at B6). Also shows that placing a tile between two segments that don't share a similar tile is also fine (Y5 at G4)
- Shows that placing a tile between two segments of different similarity types is invalid (R4 at H6), but placing a tile at the corner of two segments of a different similarity type in fine (R4 at H6)

replaceTile:

- Shows that attempting to replace a tile that does not exist in the players hand will fail (replace B2)
- Shows that attempting to replace a tile that does exist in the player's hand will succeed (replace B1)
- Shows that attempting to replace a tile when the tile bag is empty will fail (replace B2)

loadGame:

loadGame0: file does not exist
loadGame1: invalid number of players
loadGame2: invalid player name
loadGame3: invalid player score
loadGame4: invalid player hand
loadGame5: invalid player difficulty
loadGame6: invalid board format
loadGame7: invalid tilebag
loadGame8: invalid current player name
loadGame9: valid game

gameOver:

- Shows that if the tile bag is empty and once player has no tiles left in their hand, then the game will end and the winner will be named. If the scores are equal then it will result in a draw

stalemate:

- Shows that if a stalemate is reached in which no tile in any player's hand nor in the tilebag can be successfully placed, the game ends and the winner named.

Everything else:

- getInput, boardToString and displayGameState are all tested implicitly in other test cases, we will display the fact that newGame and saveGame work during the presentation.