Istanbul Technical University
Faculty of Computer and Informatics
Computer Engineering Department

BLG 458E
Homework 2 Report

Ömer Malik Kalembaşı - 150180112

January 12th, 2024

1 Explanation and how to run

1.1 Part 1 - Module Design

This section establishes the foundational structure for the chessboard. It includes data types for different chess pieces (Bishop, Knight, Rook, Queen, Pawn) and teams (Red, Green, Blue, Purple, White). The Chessboard type is defined as a two-dimensional list, with each cell possibly containing a piece and its team. This setup allows for an intuitive representation of the chessboard state.

1.2 Part 2 - Score and Change Functions

Part 2 of the chess program is dedicated to defining the movement capabilities of each chess piece. The functions involved include:

- **bishopMoves, knightMoves, rookMoves, queenMoves, pawnMoves:** Generate potential moves for each chess piece type based on their unique movement patterns.
- **validMoves:** Filters moves to ensure they are within the chessboard's bounds.
- **attackCount:** Calculates the total possible attacks on the board by evaluating each piece's attacking capabilities.
- **countAttacks and canAttacked:** Helper functions to count attacks from a specific piece and determine if a piece can attack another, respectively.
- **canAttack and related functions (pawnAttack, knightAttack, etc.):** Determine if a piece can attack a specific position, considering each piece's unique movement and attack patterns.
- **range and generatePath:** Assist in creating paths between positions and determining if they are clear of other pieces.
- **Utility functions (isPathClear, getPieceAt, etc.):** Provide support for main functions, such as checking path clearance, getting pieces at specific positions, and calculating the board's total value.

This part crucially enables the simulation of chess piece movements and interactions.

1.3 Part 3 - Chessboard Tree

Part 3 deals with exploring different chessboard states using a tree structure:

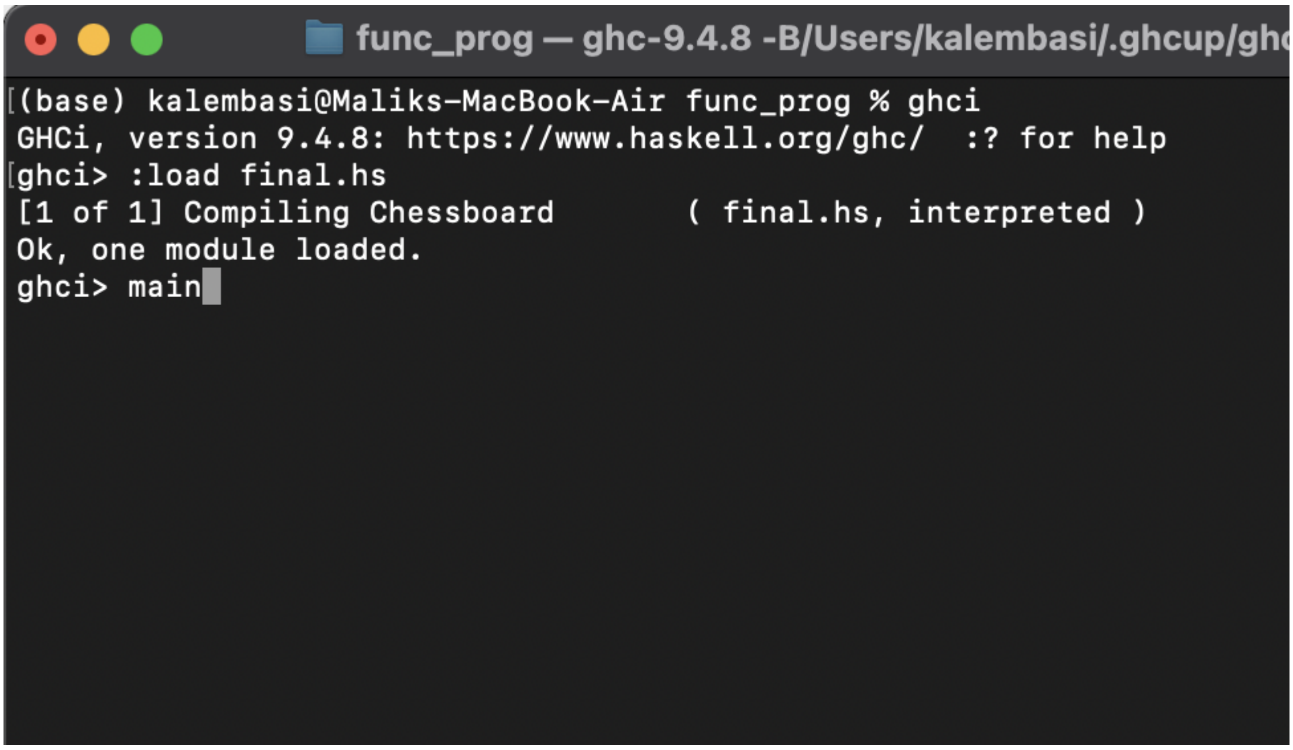
- **ChessboardTree:** This data structure represents a node in the tree, where each node is a chessboard configuration, and its children are possible configurations that can be reached from it.

- **createChessboardTree:** This function builds the chessboard tree. Given a chessboard and a depth, it creates a tree where each branch represents a potential game move. It uses functions like `horizontalCrossover`, `verticalCrossover`, and `deletePiece` to generate new board configurations.
- **findBestChessboards:** This function evaluates and finds the best chessboard configuration at each level of the tree. It uses a heuristic (`evaluateChessboard`) to determine the 'best' board, based on factors like board value and the number of attacks.

These functions collectively allow for a sophisticated analysis of chessboard states, contributing to strategic decision-making in game simulations.

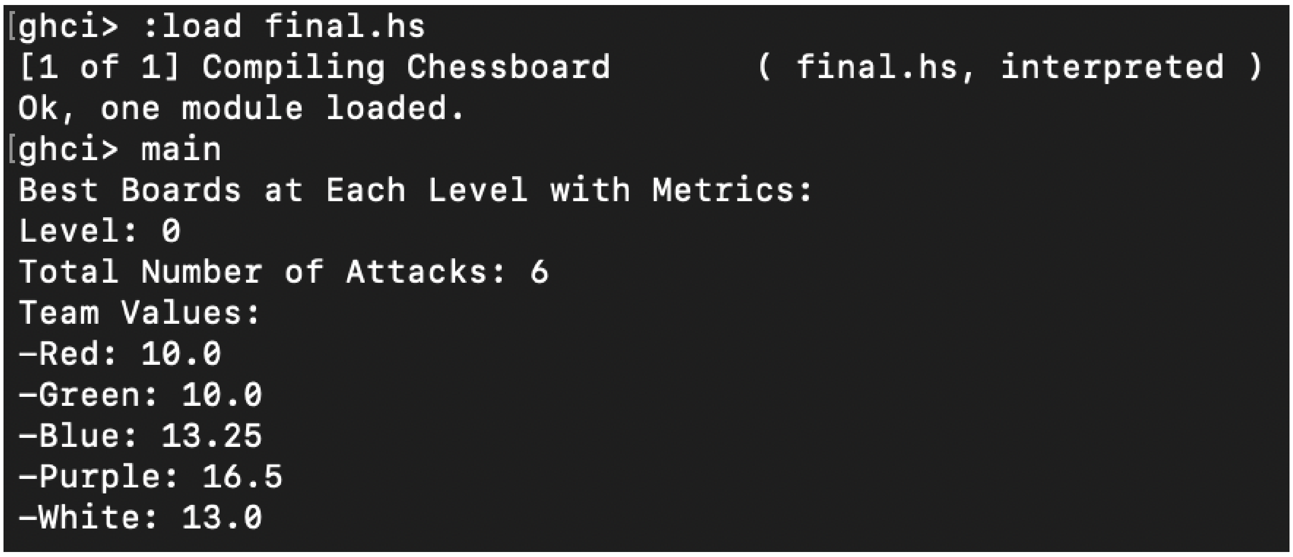
1.4 How to Run

1. Type `:load final.hs` and press enter.
2. Type `main` and press enter.



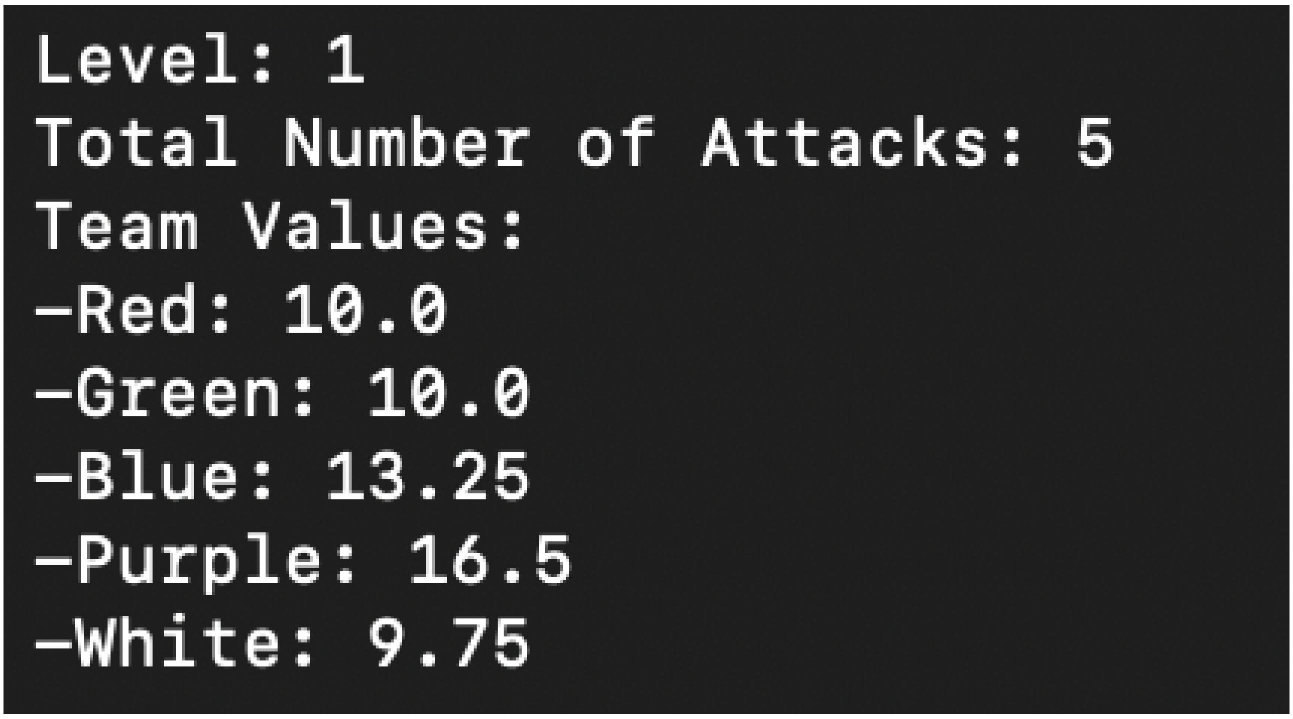
```
[(base) kalembasi@Maliks-MacBook-Air func_prog % ghci
GHCi, version 9.4.8: https://www.haskell.org/ghc/  :? for help
[ghci> :load final.hs
[1 of 1] Compiling Chessboard       ( final.hs, interpreted )
Ok, one module loaded.
ghci> main
```

Figure 1: How to run program



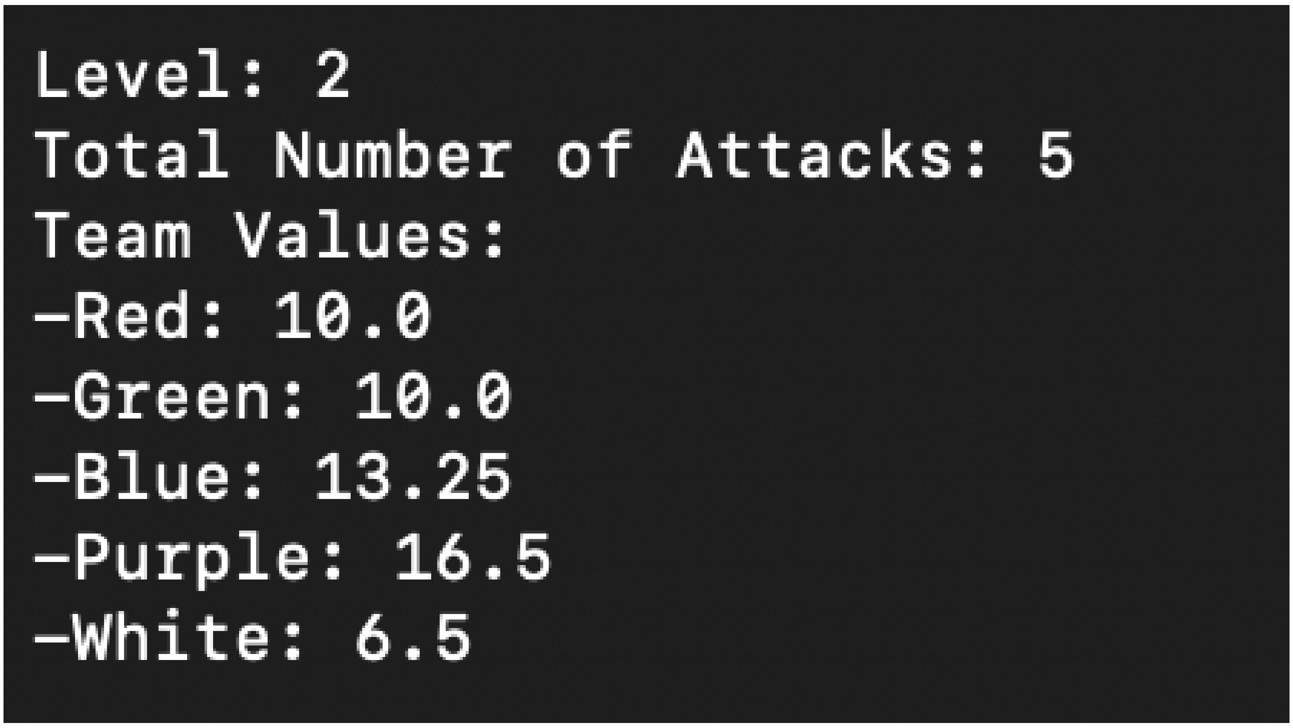
```
[ghci> :load final.hs
[1 of 1] Compiling Chessboard       ( final.hs, interpreted )
Ok, one module loaded.
[ghci> main
Best Boards at Each Level with Metrics:
Level: 0
Total Number of Attacks: 6
Team Values:
-Red: 10.0
-Green: 10.0
-Blue: 13.25
-Purple: 16.5
-White: 13.0
```

Figure 2: Level 0



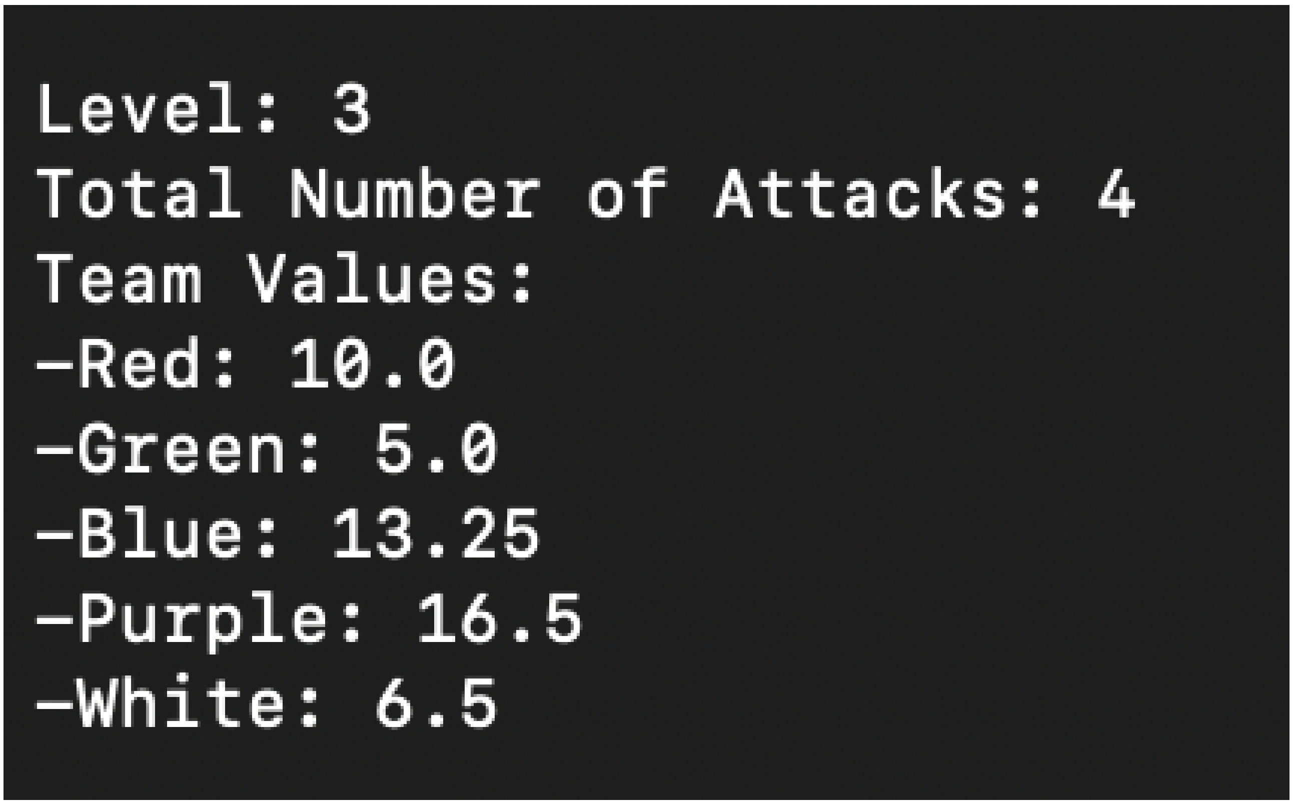
```
Level: 1
Total Number of Attacks: 5
Team Values:
-Red: 10.0
-Green: 10.0
-Blue: 13.25
-Purple: 16.5
-White: 9.75
```

Figure 3: Level 1



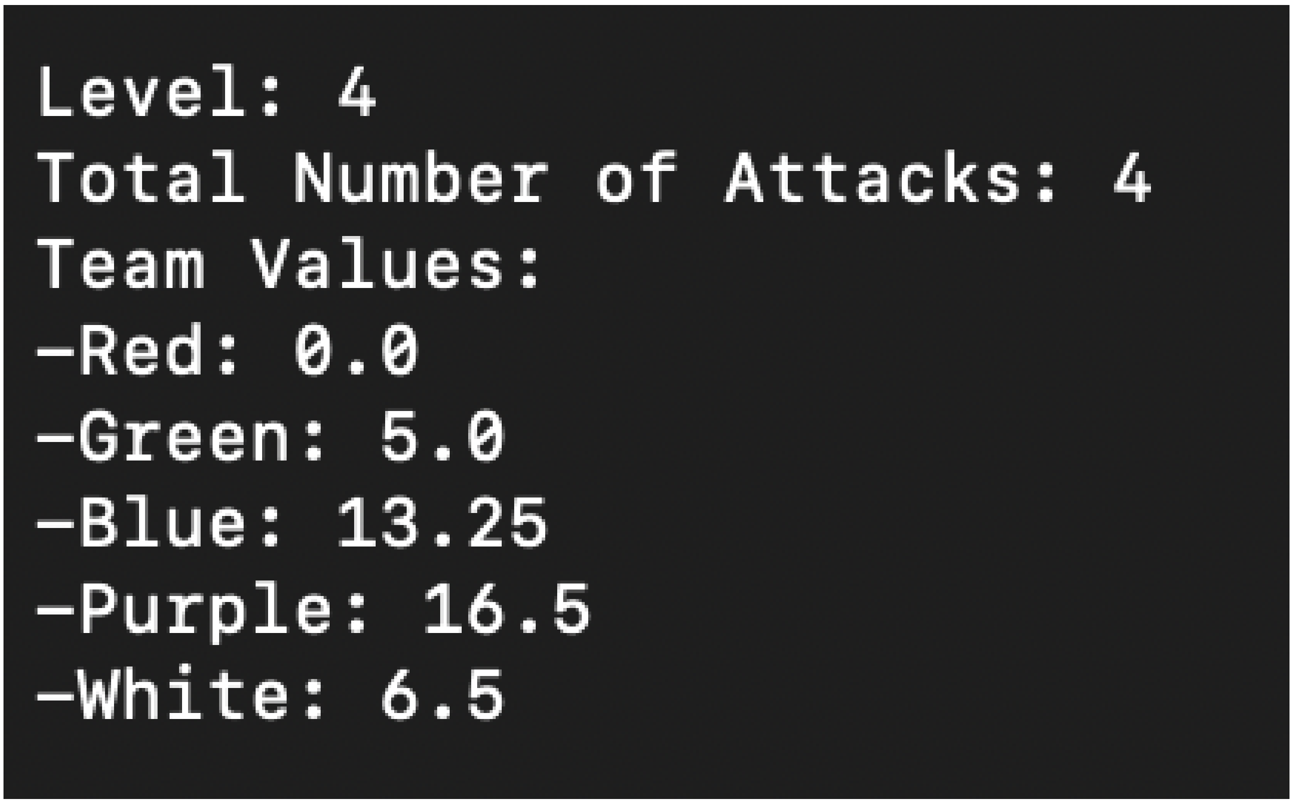
```
Level: 2
Total Number of Attacks: 5
Team Values:
-Red: 10.0
-Green: 10.0
-Blue: 13.25
-Purple: 16.5
-White: 6.5
```

Figure 4: Level 2



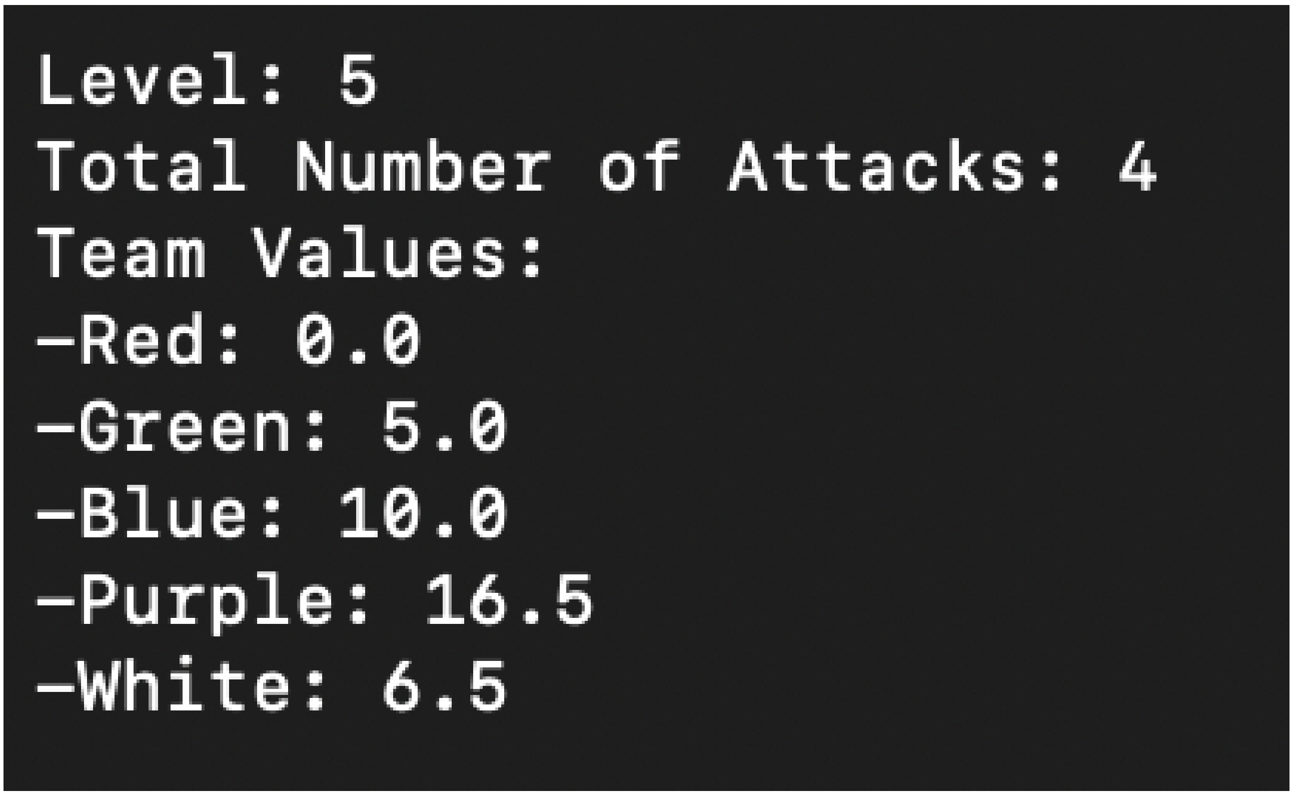
```
Level: 3
Total Number of Attacks: 4
Team Values:
-Red: 10.0
-Green: 5.0
-Blue: 13.25
-Purple: 16.5
-White: 6.5
```

Figure 5: Level 3



```
Level: 4
Total Number of Attacks: 4
Team Values:
-Red: 0.0
-Green: 5.0
-Blue: 13.25
-Purple: 16.5
-White: 6.5
```

Figure 6: Level 4



```
Level: 5
Total Number of Attacks: 4
Team Values:
-Red: 0.0
-Green: 5.0
-Blue: 10.0
-Purple: 16.5
-White: 6.5
```

Figure 7: Level 5