Assumptions

- Developers have read over the User Manual and Program Features.
- Developers are familiar with C/C++, along with the Proteus.

Important Variables Names and Uses

- **Variables for main game functionality:**
  - **status** – indicates if the game loop should continue. Default value is 1, but will be 0 if the rectangle's width is under its minimum 20 pixel size, or if the user does not land any of the block on the previous row.
  - **score** – keeps track of user's score. Calculated by multiplying the row number (plus one) by the number of pixels from current row's rectangle that landed on the previous row.
  - **difficulty** – selected from user input, it determines the speed at which the rectangle moves.
- **Variables to determine if the block's movement:**
  - **count** – incremented after the rectangle has moved its maximum distance to the right. It is modulated by two to determine if the block is moving left or right. If count % 2 equals 0, the block is moving to the right and if count % 2 does not equal two, the block is moving to the left.
  - **xPos** – the pixel value of the left most x-value of the rectangle
  - **blockTop** – the pixel value of the top most y-value of the rectangle. It uses the row number to calculate the position the rectangle should be on the screen.
- **Variables to determine what portion of the block lands on top of the previous row:**
  - **left** – indicates the current leftmost x-value for the current row's rectangle
  - **right** – indicates the current rightmost x-value for the current row's rectangle
  - **prevLeft** – indicates the leftmost x-value from the previous row
  - **prevRight** – indicates the rightmost x-position value from the previous row

Classes

- **class Game** – used to keep track of each user's stats. New class is instantiated at the beginning of each game.
  - **Private variables:** name, score, count, difficulty
  - **Public:** function game() and constructor Game()

Function Descriptions

- **void game() –** flowchart for function can be found here. This function contains all of the code for the loop that brings the user through choosing their difficulty level and playing the game. The function finishes when the user looses (variable status becomes 0) and the score for that user is updated.
- **void dispRules()** – prints all of the stacker rules to the screen when function is called.
- **int main()** contains:
  - A **welcome screen** that user taps to transition to the **main menu**, where user can choose to "play game", "view stats", "view rules", or "view acknowledgments".
  - **Play game** – instantiates the class "Game", calls function "game()", and allows user to tap screen to return to main menu.
  - **Stats** – contains all of the code that reads and writes top scores to the SD card and prints to screen. Allows user to tap screen to return to main menu.
  - **Rules** – calls function "dispRules()" and allows user to tap screen to return to main menu.
  - **Acknowledgements** – prints all acknowledgements to screen and allows user to tap screen to return to main menu.

Program Performance

- The program is not fully performing. It does not exit the game function when selected, although the majority of the functionality (as described on the Features Page) did work.

Limitations

- The game has a maximum height of six rows.