* 1. Some notable obstacles I overcame were figuring how to check for certain cases and what to return for each case. Another obstacle I overcame was fixing why my code would output “Syntax error at position 1” every time I ran it, regardless of the input. I was able to solve these problems by running through the code in my mind/on paper and thinking it through so I knew where to use the right if statement or return statement.
  2. // Include necessary header files and define global constants

// Define global constants

// Constant for horizontal direction

// Constant for vertical direction

// Constant for foreground mode

// Constant for background mode

// Function to plot a horizontal line

// Loop through columns to set characters for a horizontal line

// Function to plot a vertical line

// Loop through rows to set characters for a vertical line

// Function to plot a rectangle

// Check if height and width are valid

// If valid, call plotVerticalLine and plotHorizontalLine functions to create a rectangle

// Function to plot a line

// Check if the inputs are valid

// Depending on fgbg and direction, set characters for the line

// Return true if successful, false if unsuccessful

// Function to process horizontal part of a command

// Extract distance for a horizontal command from the command string

// Handle various cases for the distance value

// Return 0 if the distance is successfully extracted, 1 if there is a syntax error // Function to process vertical part of a command

// Extract distance for a vertical command from the command string

// Handle various cases for the distance value

// Return 0 if the distance is successfully extracted, 1 if there is a syntax error // Function to perform a series of commands

// Initialize variables to keep track of the current state

// Iterate through the command string

// Process and execute the commands based on their type (e.g., H, V, F, B, C)

// Return a status code to indicate the result of command execution (0 for success, 1 for syntax error, 2 for invalid mode or character, 3 for a command that cannot be performed) }

// Main function

// Prompt the user to input the number of grid rows and columns (max 30 each)

// Validate and set the grid size

// Initialize the current character and mode

// Enter a loop to repeatedly input and execute command strings

// Display the grid or relevant error messages based on the result of command execution

* 1. **Valid Command Sequence with Multiple Commands:**Sequence of valid commands that includes both horizontal and vertical lines, character changes, and clearing the grid: performCommands("H3V2FBG\*CH5", '#', BG, badPos);

This test case includes a sequence of commands, changing characters between FG and BG modes, and clearing the grid at the end.

* 1. **Syntax Error in Command Sequence:**Command sequence with a syntax error in one of the commands: performCommands("H3V@2FBG\*CH5", '\*', FG, badPos);

In this case, the @ in the "V@2" command is a syntax error, which should be reported.

* 1. **Invalid Mode in Command Sequence:**Command sequence with an invalid mode (not FG or BG): performCommands("H3FG@2CH5", '\*', 2, badPos);  
     In this case, the mode is set to 2, which is invalid and should be reported as an error.
  2. **Out-of-Bounds Command:**Command that attempts to draw a line that goes out of bounds: performCommands("H10FG\*CH5", '\*', FG, badPos);  
     The horizontal line in this test case attempts to go beyond the grid boundaries.
  3. **Clear Command Only:**Command sequence that only includes the clear command: performCommands("C", '\*', FG, badPos);  
     This should clear the grid without any errors.
  4. **Invalid Character in Command Sequence:**Command sequence with a non-printable character: performCommands("H3FG@2CH5", '\x07', FG, badPos);  
     The character \x07 is non-printable, and this test case should report it as an error.