**CS Report**

While the program was overall very difficult to produce, there were a few specific obstacles that prevented my code from being successful.

1. **References** – many of the functions would produce errors because of incorrect referencing (an int referencing a string, etc.). It became hard to recognize what types were designated to what variables.
2. **Many nested loops/statements** - The abundance of if-else statements in addition to the several loops confused me as I had to sort through lines and lines of code. Many of the silly errors came from if/else-if confusions and misinterpretations from some of the brackets.
3. **String to int** – It was very difficult to understand how to convert the string into an integer to use for distance. It definitely took a few hours to sort out this calculation.
4. **Following the specifics of references/commands** – Just because I had the program working doesn’t mean that it fit the requirements of the spec. It took a while to correctly plot functions without breaking the rules (changing references, putting setChar in executeCommands, etc.)
5. **Getting plotline to work on its own** – while my program did run for a long time, some of the functions were not standalone (plotline); I had to make edits so that I wouldn’t lose completeness points.

Design of the Program

Repeatedly:

Ask for a string

If the string is empty, run through executeCommands and terminate.

In executeCommands:

Repeatedly:

Check the syntax of the string

If not a valid string, return 1.

Set character position so that in the next loop it could look for a letter/command

If not, return 1;

Repeatedly:

Check if the commands can be plotted

Check if it is a command

If a directional command, get the distance, h/v (distance, dir).

Converts string distance to integer.

FGBG – get plotChar and fgbg

If clear,

Loop spaces, set ‘\*’ at (1,1)

Add change in R/C to original pt.

Try to plot the commands

If valid commands, use helper functions (plot\*\*\*)

If it is outside the grid,

return false, badPos, return 2.

If numbers aren’t designated (FG, HORIZ, etc.), return false.

Draw() if successful

Return error statement if unsuccessful

Helper functions

processHV

string distance to integer, moves position

processfb

gets FGBG, plotChar, moves position

The following is data that can be used to test if the program is working correctly:

1. **V12 vs v12:** Makes sure that uppercase and lowercase are treated the same for commands
2. **H03:** Makes sure that the 0 does not throw off the program; checks to see if the line is still plotted.
3. **H3 H2**: Checks to see if characters between the commands are truly prohibited.
4. **1H3V4**: Checks to see if invalid commands are truly prohibited.
5. **H0 or v0**: Checks if the program keeps the point, rather than disappearing.
6. **B, F, H, V by themselves**: It checks to see if syntactically it is valid.
7. **V30h, h30v, etc**.: Also checks to see if syntactically is valid before checking the string; returns 1 instead of 2;
8. **plotline versus running the whole program**: It tests for standalone calls; some things work inside the executeCommand only.
9. **F(char), B(char) by themselves**: test to make sure that they do not plot any characters on the grid.