Obstacles:

I faced many obstacles in this project. Starting off, reading the spec and digesting it was hard for me. I felt like there was so much information at once, so it was overwhelming. After reading it multiple times and doing steps at once, everything worked out. Another obstacle is when doing plotline I mixed up rows and columns with horizontal and vertical. For instance, when checking out of bounds, at first I had horizontal with rows. Tracing the code line by line helped me solve the issue. Furthermore, I also ran into the obstacle in the way I approached the code. At first, I read the string index by index to check for commands. However, for some test cases, it did not pass like h12365. This made me change my code from going through each individual index to chunk by chunk. In this case, it would go through the h chunk and see that 3 is not a valid command and then return 1. Moreover, I also learned the hard way of ASCII character to integer conversion. At first, I was not careful about changing a character to an integer. However, after some debugging, I realized not only do I have to cast it to an integer, but I also have to subtract by ‘0’.

Pseudocode:

Main is given

plotLine function

Check if it matches the basic conditions

If it's not bg, fg, horiz, vert, or printable

Return false

For horiz

Check out of bounds condition

Repeatedly

Update horizontal

If it is an fg update to distance

If it is bg and equal to ‘ ’ update to distance

Increment

For vert

Check out of bounds

Repeatedly

Update vertical

If it is an fg update to distance

If it is bg and equal to ‘ ’ update to distance

Increment

Otherwise return true

WorkingString function

Repeatedly

Check for command

Not command set badPos and return false

Check the conditions for b and f

If b is last character or not printable

Set bad post and return false

Increment

Check the conditions of h and v

If its single digit, double digit, negative single digit, or negative double digit

Increment each time

else

Set badPos and return false

If its equal to c

Increment

performFunctions

Initialize distance, currentcols, currentrows, and returnstatus

Check if it's a valid string using workingString

Return 1

Check if it’s not printable or fg and bg

Return 2

Repeatedly

If its a empty string

Return 0

If its c

Do c command

If its f

Do f

increment

If its b

Do b

increment

If its h

Check how many characters in h

Do the correct h command

Set plotline to returnstatus

If returnstatus is true

Update currentcols

Else

Set badpos and return 3

If its v

Check how many characters in v

Do the correct v command

Set plotline to returnstatus

If returnstatus is true

Update currentrows

Else

Set badpos and return 3

Increment and return 0

Test Cases for:

int performCommands(string commandString, char& plotChar, int& mode, int& badPos)

Some test cases do not have a reasoning because it falls under a reasoning stated before

1. performCommands(“”, ‘\*’, FG, 0); test empty string
2. performCommands(“h2”, ‘\*’, FG, 0); next couple to test h in all directions single digit
3. performCommands(“h12”, ‘\*’, FG, 0); double digit
4. performCommands(“h5f@h-2”, ‘\*’, FG, 0); negative single digit
5. performCommands(“h20f@h-15”, ‘\*’, FG, 0); negative double digit
6. performCommands(“H2”, ‘\*’, FG, 0); next couple are the sme as before but testing uppercase H
7. performCommands(“H12”, ‘\*’, FG, 0);
8. performCommands(“H5f@H-2”, ‘\*’, FG, 0);
9. performCommands(“H20f@H-15”, ‘\*’, FG, 0);
10. performCommands(“v2”, ‘\*’, FG, 0); Now the same for v (checking all directions)
11. performCommands(“v15”, ‘\*’, FG, 0);
12. performCommands(“v5f@v-3”, ‘\*’, FG, 0); test v negative single digit
13. performCommands(“v20f@v-15”, ‘\*’, FG, 0); test v negative double digit
14. performCommands(“V2”, ‘\*’, FG, 0); same as last 4 but checking v uppercase
15. performCommands(“V15”, ‘\*’, FG, 0);
16. performCommands(“V5f@V-3”, ‘\*’, FG, 0);
17. performCommands(“V20f@V-15”, ‘\*’, FG, 0);
18. performCommands(“f@h12”, ‘\*’, FG, 0); checking if f works
19. performCommands(“F@h12”, ‘\*’, FG, 0); checking if f uppercase works
20. performCommands(“b@h12”, ‘\*’, FG, 0); checking if b works
21. performCommands(“B@h12”, ‘\*’, FG, 0); checking if B uppercase works
22. performCommands(“h12c”, ‘\*’, FG, 0); check c
23. performCommands(“h12C”, ‘\*’, FG, 0); check uppercase c
24. performCommands(“h10v3”, ‘\*’, FG, 0); combination of h and v
25. performCommands(“h14v3c”, ‘\*’, FG, 0); start combining some commands
26. performCommands(“h5b3h2v3”, ‘\*’, FG, 0);
27. performCommands(“h”, ‘\*’, FG, 0); start checking for errors and for correct badPos
28. performCommands(“v”, ‘\*’, FG, 0); empty command
29. performCommands(“c4”, ‘\*’, FG, 0); extra number
30. performCommands(“h-”, ‘\*’, FG, 0);
31. performCommands(“v-”, ‘\*’, FG, 0);
32. performCommands(“h123”, ‘\*’, FG, 0); extra number
33. performCommands(“v123”, ‘\*’, FG, 0);
34. performCommands(“b”, ‘\*’, FG, 0);
35. performCommands(“f”, ‘\*’, FG, 0);
36. performCommands(“f@@”, ‘\*’, FG, 0); extra character without command
37. performCommands(“h35”, ‘\*’, FG, 0); check out of bounds
38. performCommands(“v35”, ‘\*’, FG, 0); check out of bounds
39. performCommands(“h-35”, ‘\*’, FG, 0); check out of bounds
40. performCommands(“v-35”, ‘\*’, FG, 0); check out of bounds
41. performCommands(“cc”, ‘\*’, FG, 0); check doubles work
42. performCommands(“ff”, ‘\*’, FG, 0);
43. performCommands(“bb”, ‘\*’, FG, 0);
44. performCommands(“hh”, ‘\*’, FG, 0); shouldnt work doubles
45. performCommands(“vv”, ‘\*’, FG, 0);
46. performCommands(“v50f”, ‘\*’, FG, 0); testing what badpos comes out should be 4 because syntax erros before plotting errors
47. performCommands(“h20v15b@h3v-3f$v1”, ‘\*’, FG, 0); combination of different commands trying different direction
48. performCommands(“h14v3h3c”, ‘\*’, FG, 0); see if clear works over others
49. performCommands(“#”, ‘\*’, FG, 0); check random characters at the start
50. performCommands(“v12#h2”, ‘\*’, FG, 0); random characters in between
51. performCommands(“v12h2#”, ‘\*’, FG, 0); random characters at the end
52. performCommands(“h5f h-4”, ‘\*’, FG, 0); check if spaces work with f
53. performCommands(“h5b4h-4”, ‘\*’, FG, 0); check b with something filled
54. performCommands(“v12#h123”, ‘\*’, FG, 0); should be leftmost error
55. performCommands(“h23c h”, ‘\*’, FG, 0); check correct error position after c
56. performCommands(“f”, ‘\n’, FG, 0); check valid character
57. performCommands(“h 5”, ‘\*’, FG, 0); see if error with extra space

Test for bool plotLine(int r, int c, int distance, int dir, char plotChar, int fgbg);

Besides the given test

1. plotLine(1,1,35, HORIZ, ‘\*’, FG) checking out of bounds
2. plotLine(1,1,35, VERT, ‘\*’, FG) checking out of bounds
3. plotLine(1,1,3, VERT, ‘\*’, FG) check if its normal
4. plotLine(1,1,3, HORIZ, ‘\*’, FG) checking normal