Project 3 Report

1. A brief description of notable obstacles you overcame.
   1. Handling the character that comes after the ‘H’ /’V’ command: There are multiple situations: 1. One number and the string of commands ends. 2. One number followed by another command. 3. Two numbers followed by another command. 4. Previous situations, but with a ‘-’ sign. When I’m handling the first number, I must determine whether the string of commands have ended or not after that number. If so, I must not try to access the character (which does not appear) after the last number, or a runtime error will appear. To overcome this obstacle, I used if(i!=commandString.size()-1) to determine whether the number is the last character of the command string, and handle different situations differently.
   2. I realized that I must check for the syntax of the string of commands before running it, so I implemented a function bool checkCommands(string commandString, int& badPos) that looks for syntax errors before running the program.
   3. During Phase 2, I found my code that checks the input for fgbg/vertical or horizontal distance/characters became long and very hard to read. Therefore, I implemented seperate functions such as ' validHorizDistance(**int** c, **int** distance)’ or ‘validFgbg(**int** fgbg)’, and call them in my plotLine() function in Phase 2, to make my program more abstract and easier to understand/debug.
2. Description of program design

//plotline function:

If direction is HORIZ:

if distance is valid AND char is valid AND fgbg is valid:

if fg:

repeat until all character is looped:

plot a char;

if bg:

repeat until all characer is looped:

if nextChar is ‘ ’:

plot a char;

else:

break;

If direction is VERT:

if distance is valid AND char is valid AND fgbg is valid:

if fg:

repeat until all character is looped:

plot a char;

if bg:

repeat until all characer is looped:

if nextChar is ‘ ’:

plot a char;

else:

break;

//End of plot line function

//start of Phase 3

if invalidChar or invalidMode:

break;

bool checkCommands(string commandString, int&badPos):

repeatedly: //Only Checks for syntax error

find next char:

if next char is ‘C’ or ‘c’:

go to next char;

else if next char is ‘F’ or ‘B’ or ‘f’ or ‘b’:

if next char is not printable:

set badPos;

break;

else if next char is ‘H’ or ‘h’ or ‘V’ or ‘v’:

if nextChar is NOT number:

if nextChar is NOT ‘-‘:

set badPos;

break;

else:

proceed to nextChar;

else:

set badPos;

break;

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

if checkCommands is not true:

break;

repeatedly: //Plot the graph

find next char:

if next char is ‘C’ or ‘c’:

clearsGrid, resets penLocation, mode, character.

go to next char;

else if next char is ‘F’ or ‘f’:

if next char is valid:

set mode to FG;

sets currentChar to next char;

go to next char;

else:

break;

else if next char is ‘B’ or ‘b’:

if next char is valid:

set mode to BG;

sets currentChar to next char;

go to next char;

else:

break;

else if next char is ‘H’ or ‘h’:

if next char is number:

proceed to nextChar;

if nextChar is number:

proceed to nextChar;

concatenate the two characters to a int;

else:

concatenate the character to a int;

else if next char is ’-’:

proceed to the char after next char;

if nextChar is number:

proceed to nextChar;

concatenate the two characters to a int;

else:

concatenate the character to a int;

if the horizontal distance is valid:

call plotline function

update currentRow & currentCol;

else:

update badPos;

else:

update badPos;

break;

else if next char is ‘V’ or ‘v’:

if next char is number:

proceed to nextChar;

if nextChar is number:

proceed to nextChar;

concatenate the two characters to a int;

else:

concatenate the character to a int;

else if next char is ’-’:

proceed to the char after next char;

if nextChar is number:

proceed to nextChar;

concatenate the two characters to a int;

else:

concatenate the character to a int;

concatenate the characters to a int

if the vertical distance is valid:

call plotline function

update currentRow & currentCol;

else:

update badPos;

else:

update badPos;

break;

else:

update badPos;

break;

1. Test data used: (Assume 20\*20 grid is set up)
   1. Testing the performCommands function:
      1. F@H2 [Test changing character command – Upper Case]
      2. f@H2 [Test changing character command – Lower Case]
      3. FxH3B@H5h-7 [Test Background command -Upper Case]
      4. C [Test clear screen command]
      5. Fxh3B@H5h-7 [Test Background command -Lower Case]
      6. c [Test clear screen command -lower case]
      7. H12h3Fxh-10H-1 [Test for 2 Digit, 1 Digit, 2 Digit with negative Sign, 1 digit with negative sign for command ‘h’ and ‘H’]
      8. v3V11FtV-10v-1 [Test for 2 Digit, 1 Digit, 2 Digit with negative Sign, 1 digit with negative sign for command ‘v’ and ‘V’]
      9. A3 [Test for syntax error command – expected ‘Syntax error at position 0’]
      10. H20F [Test for syntax error– expected ‘Syntax error at position 4’]
      11. fgh [Test for syntax error command – expected ‘Syntax error at position 3’]
      12. v-\* [Test for syntax error – expected ‘Syntax error at position 2’]
      13. v-1\* [Test for syntax error – expected ‘Syntax error at position 3’]
      14. v-\*1 [Test for syntax error – expected ‘Syntax error at position 2’]
      15. v- [Test for syntax error – expected ‘Syntax error at position 2’]
      16. fghxv3 [Test for syntax error command – expected ‘Syntax error at position 3’]
      17. H20 [Test for out of bound commands - double digit number – expected ‘Cannot perform command at position 0’]
      18. H18h2 [Test for out of bound commands - single digit number – expected ‘Cannot perform command at position 3’]
      19. H-5 [Test for out of bound commands - single digit number with negative sign – expected ‘Cannot perform command at position 0’]
      20. h-20 [Test for out of bound commands - double digit number with negative sign – expected ‘Cannot perform command at position 0’]
      21. h5 fgv3 [Test for unexpected blank space in command – expected Syntax error at position 2]
      22. h5f v3 [Test for valid blank space after ‘f’ command]
   2. Testing the plotline function:
      1. plotLine(5,5,3,HORIZ,'\*',FG); [Test for horizontal drawing-foreground]
      2. plotLine(5,5,3,VERT,'\*',FG); [Test for horizontal drawing-foreground]
      3. plotLine(5,5,10,HORIZ,'%',BG); [Test for horizontal drawing-background]
      4. plotLine(5,5,10,VERT,'%',BG); [Test for vertical drawing-background]
      5. plotLine(15,15,-10,VERT,'%',FG); [Test for vertical drawing- negative direction]
      6. plotLine(15,15,-10,HORIZ,'%',FG); [Test for vertical drawing-negative direction]
      7. plotLine(15,15,6, HORIZ,'%',FG); [Test for out-of-plot command]
      8. plotLine(15,15,6, VERT,'%',FG); [Test for out-of-plot command]
      9. plotLine(1,1,-10, HORIZ,'%',FG); [Test for out-of-plot command]
      10. plotLine(1,1,-10, VERT,'%',FG); [Test for out-of-plot command]
      11. plotLine(1,1,5, VERT,'\n',FG); [Test for unprintable character]
      12. plotLine(1,1,5, VERT,'\t',FG); [Another test for unprintable character]
      13. plotLine(5,5,3,25,'\*',FG); [Invalid parameter for direction (not VERT or HORIZ)]
      14. plotLine(5,5,3,VERT,'\*',-1); [Invalid parameter for fgbg (not FG or BG)