CS/CE 1337 – PROJECT 3 – Super Mario Paint

Pseudocode Due: 3/14 by 11:59 PM

Project Due: 4/4 by 11:59 PM

KEY ITEMS: Key items are marked in red. Failure to include or complete key items will incur additional deductions as noted beside the item.

Submission and Grading:

- All project deliverables are to be submitted in eLearning.
- The pseudocode should be submitted as a Word or PDF document and is not accepted late.
- Zip all of the source files into a single zipped file
    - Make sure the zipped file has a .zip extension (not .tar, .rar, .7z, etc.) (-5 points)
    - Please review the submission testing information in eLearning on the Course Homepage
- Projects submitted after the due date are subject to the late penalties described in the syllabus.
- Programs must compile using gcc 7.3.0 or higher with the following flags enabled
    - -Wall
    - -Wextra
    - -Wuninitialized
    - -pedantic-errors
    - -Wconversion
- Each submitted program will be graded with the rubric provided in eLearning as well as a set of test cases. These test cases will be posted in eLearning after the due date. Each student is responsible for developing sample test cases to ensure the program works as expected.
- **Type your name and netID in the comments at the top of all files submitted. (-5 points)**

Objectives:

- Use structures containing pointers to create a dynamic data structure.
- Use pointers to modify the dynamic data structure.
- Implement a recursive function.

Problem: Mario Paint was a great innovation in its time allowing people with a home console to create art and music easily. One of the features of Mario Paint was that it let users create their own 8-bit stamps to use within the game. With this idea, you are going to create a program that will allow users to create black and white pixel art. Since all of the output will be file-based, we can't easily add color to the drawings.

Pseudocode: Your pseudocode should describe the following items

- Creating the grid
- Printing the grid
- Deleting the grid
- For each function
    - Determine the parameters and return type
    - Detail the step-by-step logic that the function will perform

Details:

- The user's canvas will be a grid of node structures (50 rows of 50 nodes)
- **Node structure**
  - Variable to hold character for drawing
  - 4 node pointers
    - Up, down, left and right
- The nodes will be connected to other nodes through the pointers inside them.
- The canvas will be constructed and connected without the help of any other data structures or arrays (-10 points if arrays/data structures used to help connect nodes)
- There will be a distinct border for the canvas indicated by null pointers
  - A node on the top border of the canvas will have a null up pointer
  - There will not be any wraparound effect.
- Input will consist of a series of commands read from a file
- Each command will indicate pen status as well as movement direction and distance
  - Optionally, a command may also include a bold status to include a different character when drawing
- The pen status will determine if the pen is on the paper or in the air
- If the pen is "down" and moved, write the given number of characters in the file in the given direction
- If the pen is "up", just move the file pointer to the proper position
- The movements do not include the current spot of the pen
  - For example if the pen is down, the command 3, E, 2 would draw in the two spaces to the right of the current pen location.
- In the case of intersecting lines, bold will always take precedence.
- All "drawing" is to be written to the canvas in memory
  - Do not use a 2-dimensional array to hold the drawing
    (-20 points for holding/manipulating the drawing in a 2D array or using random file access techniques)
- The character to draw in the canvas will be an asterisk (*)
  - If the bold status is on, the character to draw is a hashtag (#)
- A single recursive function will be used to display the canvas (either to console or output file)
  (-10 points if done iteratively)
- The pen will begin at row 1, column 1 at the start of the program.

Input: All input will be read from a file named *commands.txt*. Each command will be on a separate line in the file and each line will have a new line character at the end of the line (except for the last line which may or may not have a newline character).

Each valid command will have the following format (note there are no spaces in the command):

<status>,<direction>,<distance>,<bold - optional>,<print - optional> - each element will be a single character

- Status
  - 1 – pen is up
  - 2 – pen is down

- Direction
    - N – north/up
    - E – east/right
    - S – south/down
    - W – west/left
- Distance – a positive integer value
- Bold - a capital B
    - This element is only valid if the pen is down
    - This element is optional and can be omitted from a valid command
- Print – a capital P
    - Print the current status of the drawing to the screen
    - This element is optional and can be omitted from a valid command

Examples of valid commands

- 1,N,5
- 2,E,2
- 2,S,3,B
- 1,W,4,4
- 2,N,10,B,4

It is possible for the file to have invalid commands. An invalid command is any command that does not adhere to the format listed above. It is also possible for a command to move the pen out of bounds of the grid. These commands should also be treated as an invalid.

If an invalid command is identified, do not execute it. Move to the next command in the file.

Output: Each input file will produce an output file named *paint.txt*. If a print option is included with the command in the input file, display the current state of the canvas to the console window using the recursive function you created. Make sure that each row of the canvas is displayed on a separate line in the console window. After the canvas has been displayed to the console, send two blank lines to the console window as a buffer for the next print option in the file.

At the end of the program, write the contents of the canvas to the output file using the recursive function.