CS/CE 1337 – PROJECT 1 – Mario Paint

Pseudocode Due By: 8/29 by 11:59 PM (No late submission)

Implementation Due By: 9/5 by 11:59 PM (No late submission)

Final Code Due By: 9/12 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:

- The pseudocode will be submitted in eLearning as a Word or PDF document and is not accepted late.
- All project source code will be submitted in zyLabs.
  - 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
- **Type your name and netID in the comments at the top of all files submitted. (-5 points)**

Objective:

- Directly manipulate files using advanced file functions
- Utilize string and character functions to perform basic input validation

Problem: This year Mario is celebrating his 35th birthday. Back in the days of the Super Nintendo console, Nintendo released a game named Mario Paint that helped people create art and music easily. To commemorate the occasion, you are going to create a program that will allow users to create black and white ASCII art.

Pseudocode: Your pseudocode should describe the following items

- Main.cpp
  - Detail the step-by-step logic of the main function
  - List functions you plan to create
    - Determine the parameters
    - Determine the return type
    - Detail the step-by-step logic that the function will perform

Implementation Phase:

- Process the sample input file and generate the sample output shown
- No invalid input
- No boundary checks

Details:

- Use the template posted in ZyLabs.
    - The code in the template copies `paint_base.txt` to `paint.txt`
        - This copy is necessary because ZyLabs will not allow the original file to be modified
    - `paint.txt` is the file that will be modified for the entirety of the program
- The user's canvas will be a grid (50 lines with 50 characters on each line)
    - The newline character on each line is not considered to be part of the canvas
    - `paint.txt` will initially consist of 50 lines with 50 space characters on each line
- Input will consist of a series of commands read from a file
    - Prompt the user for the name of the input 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 done directly in the file
    - Do not use a 2-dimensional array to hold the drawing and output it to the file at the end of the program. (-20 points for holding/manipulating the drawing in memory)
- The character to draw in the file will be an asterisk (*)
    - If the bold status is on, the character to draw is a hashtag (#)
- The pen will begin at row 1, column 1 at the start of the program.

Input: All input will be read from a file. Prompt the user for the name of the input file. 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,P
- 2,N,10,B,P

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: All output will be sent to a file named **`paint.txt`**. If a print option is included with the command in the input file, display the current state of the entire file to the console window, making sure that each line of the output file is displayed on a separate line in the console window. After the file 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.