CSCI-1200 Data Structures — Fall 2017
Homework 1 — ASCII Font Art

Before starting this homework, make sure you have read and understood the Academic Integrity Policy.

In this homework you will work with command line arguments, file input and output, and the C++ STL `string` and `vector` classes to create "ASCII Font Art" from simple input text messages. For extra credit, your program will also be able to reverse the process, and transform ASCII font art back to the original simple text string. Please read the entire handout for the assignment before starting to program. In addition to the lecture notes, you will also want to refer to the "Misc. C++ Programming Information" and "Homework Grading Criteria" sections of the course webpage.

```
@....@........@......@.....................@....@...............@..........@..@......
@....@..@@@...@......@.......@@@...........@....@..@@@....@@@...@..........@..@......
@@@@@@.@...@..@......@......@...@..........@....@.@...@..@......@.......@@@@..@......
@....@.@@@@...@......@......@...@..........@....@.@...@..@......@......@...@..@......
@....@.@......@......@......@...@..........@.@@.@.@...@..@......@......@...@..........
@....@..@@@@...@@.....@@.....@@@............@..@...@@@...@.......@@.....@@@@..@......
.....................................................................................
```

Command Line Arguments

To create an ASCII font art message, your program will expect 5 command line arguments. The first argument is the string "`display`". The second is the filename containing the bitmap font that will be used to render each character of the message in large format. The third argument is the message, a string of one or more characters. The fourth argument is a character that is to be used as the foreground (the letters). And the fifth argument is a character that will be used as the background for the ASCII font art. To create the output above, your program will be called with this command line:

```
./ascii_font_art.out display simple_font.txt Hello\ World! @ .
```

Note that the backslash is used to include a space in the message (without it the program would see 6 command line arguments). You may also use single ' or double " quotes to delimit the message or when using special foreground and background characters (see the next example below). The program will output the ASCII font art message to the terminal (to `std::cout`). Alternatively, you can use command line *file re-direction* to send `std::cout` to a file:

```
./ascii_font_art.out display simple_font.txt 'Hello World!' '*' " " > hello_world_output2.txt
```

You must exactly follow the specifications for the command line and output to ensure you receive full credit for your work. We have provided sample output files on the course website, and the validation script on Submitty will also help you check your work.

You should implement simple error checking to ensure that the arguments provided are appropriate. Your program should exit gracefully with a useful error message sent to `std::cerr` if there is a problem with the arguments.

Font File Format & Parsing of the Font File

The `simple_font.txt` file contains a fixed width "bitmap" representation of each printable character. In this file every letter uses '`#`' for the foreground character and '`.`' for the background. You'll need to swap in the desired foreground & background characters as you output the artwork.

We have provided code to help your program load the font file. Please review this code carefully. You may use/modify this code as you wish in your solution.

Note on Viewing ASCII Font Art Output & Output Files

The output of this homework will be wide. You may need to full screen your terminal (or file viewer) and decrease your font size to see the whole thing. If you don't, the individual lines of the output will probably wrap and the message will be jumbled.

Also, make sure you're using a good file viewer/editor to look at these files. It should correctly display the UNIX/GNU Linux '`\n`' line ending. Use one of the "Plaintext & Code Viewers/Editors" listed on the "C++ Development" page. Don't attempt use the Windows line ending character '`\m`' or '`\r`' because this will fail validation tests on Submitty.

Extra Credit: Reversing the ASCII Font Art

If the user specifies "`read`" mode on the command line instead of "`display`", with a total of 3 command line arguments, this will indicate that your program should reverse the process. The second argument is again the filename of the font and the third argument is the name of a file containing the ASCII font art message that should be decoded. For example:

```
./ascii_font_art.out read simple_font.txt hello_world_output2.txt
```

Your program will study the artwork in the `hello_world_output2.txt` file, breaking it up into blocks with width equal to the `simple_font.txt` width, and comparing each block to the letters from the font file. Your program will then output the encoded message to the `std::cout`.

```
Hello World!
```

Submission Details

Do all of your work in a new folder inside of your Data Structures homeworks directory. You should use the C++ STL `string` and `vector` classes in your implementation. Use good coding style when you design and implement your program. Review the "Homework Grading Criteria" section on the course webpage to be sure that the TAs will be able give you credit for your hard work. Organize your program into functions: don't put *all* the code in `main`! Use good variable and function names. Be sure to make up new test cases and don't forget to comment your code!

Download and fill out the provided template `README.txt` file, adding any notes you want the grader to read. You must do this assignment on your own, as described in the "Collaboration Policy & Academic Integrity" handout. If you discuss the problem or error messages, etc. with anyone, please list their names in your `README.txt` file. Prepare and submit your assignment as instructed on the course webpage. Please ask a TA if you need help preparing your assignment for submission or if you have difficulty writing portable code.