Cryptogram Part 3

This assignment is extremely important – (nearly) every assignment after this one uses this one!

If you have bugs or missing features in this, you will need to fix them before you can continue on to new assignments. This is very typical in software development outside of school.

You must submit .c files. Any other file type will be ignored. Especially ".o" files.

You must not zip or otherwise compress your assignment. Brightspace will allow you to submit multiple files.

You must submit buildable .c files for credit.

Introduction

There are a few things missing from our game. One is to have a large selection of phrases to decode instead of just one. Another is to detect when the game is complete. Finally, we will time how long solving the puzzle takes and print it when the game is complete.

We will source our phrases from:
https://gist.githubusercontent.com/erickedji/68802/raw/7264f2d232702b4013490a0b2f9286cfa1b817e3/quotes.txt

In case the file disappears, I will include a copy with the assignment. Take a look at this file in your text editor. Notice a few things:

1) Each quote may span several lines.
2) Each quote ends with "-- Author" and then a blank line.

We will read the entire file into memory, then randomly select a quote. The quote will be stored in two parts – the quote itself and then the author. We will merge the lines of the quote into a single string but one that contains a line feed so that we can print it nicely formatted.

To detect if the game is complete, we will decode the player's decoded version with the original.

We will give the player a chance to start a new game or to quit. Of course, the player can quit any time by typing "quit" as we implemented previously.

As with the previous assignment, the data structure is critical here. We will build a linked list of structures of "quote". A quote holds 3 things – a pointer to the phrase string, a pointer to the author string and a pointer to the next quote.

Details

Create the quote structure. Create a method that uses malloc() to allocate a quote structure and set each field to NULL.

Create a new method to loadPuzzles(). This method will read each line from the input file using fgets(). For each line, it has to do one of three things:

1) If the line is less than three characters, that means that it is blank (\r or \r\n mean a blank line is 1 or 2 characters). When this happens, it means that we are done with this quote entry, so make a new one and append the old one to the linked list.
2) If the line starts with "—", then this is an author. Make a string for that information and put it in the current quote's "author" string.
3) Otherwise, this is part of the phrase. Two possibilities here – this is the first line or it is the second or subsequent line.
    a. If it is the first line, allocate space to match the read line, copy the string into that space and set that space to be the current quote's phrase.
    b. Otherwise, allocate space for the existing string PLUS the read line. Copy the existing string into the space, then concatenate the read line. Set the current quote's phrase to this new string. Don't forget to free the old space!

loadPuzzles() should keep track of how many entries it makes and store that as a global variable.

getPuzzle() will change – it the quote count is 0, it means that we haven't loaded them yet, so call loadPuzzles(). Additionally, it should pick a random quote (remember we know how many there are). It should loop over the linked list to get to that quote and return it.

displayWorld(), oddly, is a convenient place to see if the puzzle is decoded. As you are printing the decoded version, you can check to see if there are characters not decoded OR any that don't match. If that is the case for ANY character, the player is not done. Modify displayWorld() to return that information.

Make a new function to free the loaded quotes and the whole linked list structure.

Finally, modify main a little. Main should call initialize(), gameloop() and teardown() then ask the user if they would like to play again. If they answer no, then free all of the quotes and exit.

My code was around 200 lines. Yours shouldn't be much longer or shorter.

Your program **must** run through Valgrind without any memory leaks. Any memory leaks or negative Valgrind messages will result in significant penalty, regardless of the rubric below.

Rubric	Poor	OK	Good	Great
Comments	None/Excessive (0)	“What” not “Why”, few (5)	Some “what” comments or missing some (7)	Anything not obvious has reasoning (10)
Variable/Function naming	Single letters everywhere (0)	Lots of abbreviations (5)	Full words most of the time (8)	Full words, descriptive (10)
Structure	None of: indentation matches braces {}, helper functions (0)		One of: indentation matches braces {}, helper functions (5)	Both of: indentation matches braces {}, helper functions (10)
Create quote structure	Missing (0)			Exists, holds 3 pointers to appropriate data types (5)
Quote allocator	Missing (0)			Allocates space, uses sizeof(), sets pointers to NULL (5)
loadPuzzles – files	Missing(0)		Two of: Uses fgets(), reads each line, closes file. (5)	Uses fgets(), reads each line, closes file. (10)
loadPuzzles – memory	Not dynamic (0)			Uses per-line dynamic memory, freeing unused memory (10)
loadPuzzles – logic	Missing (0)			Decides on a per-line basis what type of line this is and takes appropriate action (10)
loadPuzzles – strings	Uses ad-hoc string management (0)			Uses standard C library functionality (5)
getPuzzle	No changes(0)		One of: loads puzzles one time, gets a random puzzle (3)	All of: loads puzzles one time, gets a random puzzle (5)
displayWorld	No changes(0)			Returns if puzzle is complete (5)
teardownAll	Missing (0)			Frees all memory (10)
Main/gameloop	No changes(0)			Allows multiple games (5)