Cryptogram Part 1

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.

<u>You must submit buildable .c files for credit.</u>

Introduction

Throughout this semester, you will be building an interactive cryptogram game. Cryptograms are very simple. The player is presented with an encrypted string and must decode it.

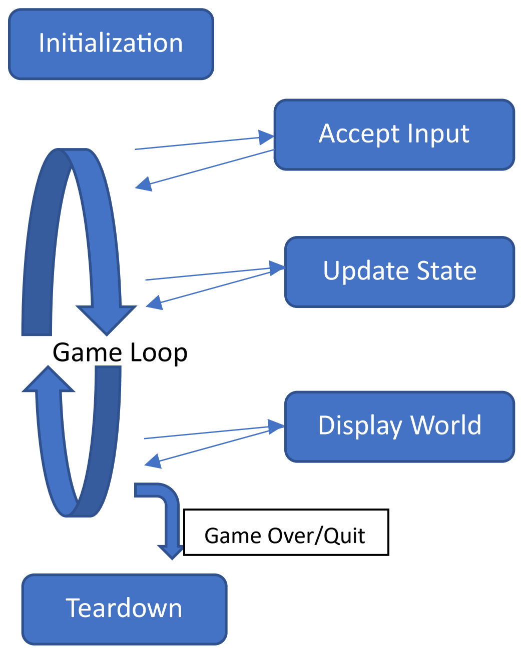
Constructing a cryptogram is very straightforward. We make a chart of the letters and then randomly choose a letter to replace that letter in the encrypted version.

Example:

A → Q
B → C
C → T (etc)

The letters on the left side are in order from A-Z. The letters on the right side are in random order and are not repeated.

Most computer games have a particular structure, whether "guess a number" or Battlefront II. They start with some initialization – set up the data structures that model the "game world". Then there is the "game loop" – a (nearly) infinite loop that accepts user input, updates the state of the game world, and updates the display. Finally, there is the teardown sequence that happens at game exit.



Details

Start a new C program.

Create a main() that calls a function called initialization(), then gameLoop(), then teardown(). Should return 0.

Initialization will call a function (getPuzzle()) which will return a string for a new puzzle. You can make getPuzzle() return a constant string of your choice – any sentence. Initialization() will set a global variable to the string.

gameLoop() will loop over the following functions:

acceptInput() will print a prompt:

Enter a letter and then the letter to replace it with or 'quit' to quit.

Then accept a line from the user (use “fgets()”). Return the string.

updateState() will take the line from acceptInput(). It will return true if the string is “quit” OR is NULL and false otherwise.

displayWorld() will print the global puzzle variable as a string.

gameLoop() will stop looping when updateState() returns true (it is OK to call displayWorld() one more time).

tearDown() will print “All Done”.

Some common pitfalls:

1) In C, you have to either declare a function before you use it OR create a forward declaration.
2) Don’t forget to #include – fgets is in stdio.h, string functions are in string.h and Booleans are in stdbool.h
3) fgets takes a few parameters – the string to fill, the size of the string (use sizeof()) and where to read from. In our case, we are reading from **stdin**.
4) fgets() returns the read in line, including the carriage return.
5) I am referring to every function like this: someFunction(). You need to determine the parameters and the return type.

Hints:

Start early. Don’t wait until it is due – you won’t have time to ask questions or get help.

Don’t write too much code – my solution is about 50 lines of code. If you go TOO much over that, you have done it really wrong. Shorter isn’t better, but longer isn’t better either.

Write extra helper functions if you find yourself copy/pasting code.

One mistake that people often make is to have AcceptInput call UpdateWorld which then calls DisplayState. This is not correct. Each of these functions are called by the game loop.

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)
Initialization	Doesn’t exist or is not called (0)			Exists, is called (5)
Teardown	Doesn’t exist (0)			Exists, is called (5)
getPuzzle	Doesn’t exist (0)			Exists, is called, returns a string (5)
Game Loop Exit	Doesn’t exist/Empty (0)	Exists, doesn’t end (3)	Exists, ends using break (7)	Exists, ends without break (10)
Game Loop Design	Doesn’t Exist/Empty	Calls some of the functions (5)		Calls all three functions (10)
Input Handling	Doesn’t exist/Empty (0)	Prompt OR accepts input (5)		Prompt and accepts input (10)
Display	Doesn’t exist/Empty (0)		N/A	Prints string (10)
Update World	Doesn’t exist/Empty (0)	Accepts Input parameter (3)	Two of: Accepts Input parameter, detects quit, detects null properly (7)	Three of: Accepts Input parameter, detects quit, detects null properly (10)
Main	Doesn’t exist (0)			Exists and returns an exit value(5)