# Words Without Friends 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 submit buildable .c files for credit.***

A black screen with white text

Description automatically generated

## Introduction

Throughout this semester, you will be building a word scramble game – the game will present the player with a list of letters and the player must find English words that can be made from those letters. Here is an example:

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.

Initialization

Accept Input

Update State

Game Loop

Display World

Teardown

Game Over/Quit

## Note – for this assignment, we will “merge” AcceptInput and UpdateState into the same function. Sometimes that makes sense. The game loop design pattern is flexible.

## Details

Start a new C program.

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

Initialization will eventually load the dictionary into memory. For now, it needs only to initialize the random number generator:

srand(time(NULL));

Initialization() will return a number – the count of words in the dictionary. For now, return 0.

gameLoop() will loop over the following functions:

displayWorld() will print the puzzle. For now, print only a line of hyphens.

acceptInput() will print a prompt: “Enter a guess: “

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

Capitalize the input from the user and print it. Make sure to strip off carriage returns and line feeds.

gameLoop() will stop looping when a function called “isDone” returns true.

isDone for now should always return true.

tearDown() will print “All Done”.

In addition, we will write two support functions that we will need later.

When we choose a set of letters, we will need to check each word in the dictionary to see if that word can be made up with the letters we have chosen.

For example, consider the letters from our example above - A D E F N O R R S S W

The first word in the dictionary, “A” – can it be made up from those letters? Yes!

The second word in the dictionary, “AARDVARK” – can it be made up from those letters? No!

This might seem like an easy algorithm – take each letter in the dictionary word and look for it in the list of letters. That won’t work! Why? It doesn’t properly account for repeated letters. ADD, for example, would show up as a valid word.

We will do something a little clever – we will make a function that, given a word and an array of 26 integers (hmm – why 26?), we will count the number of instances of that letter in the incoming word. For ADD the first few entries in the array would be: 1,0,0,2 (1 A and 2 D’s).

Let’s call this “getLetterDistribution()”.

Now, let’s make a function that compares two distributions. The idea is that we have 1 distribution that represents our choices ( A D E F N O R R S S W) and another distribution that matches our candidate word (like “ADD”). This function, compareCounts(), returns true if the candidate word can be made from the choices.

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 and line feed.
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 DisplayWorld. 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) |
| getLetterDistribution | Doesn’t exist (0) | Attempted (5) |  | Exists and is correct (10) |
| 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 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 (5) |
| compareCounts |  | Attempted (5) |  | Exists and is correct (10) |
| Main | Doesn’t exist (0) |  |  | Exists and returns an exit value(5) |