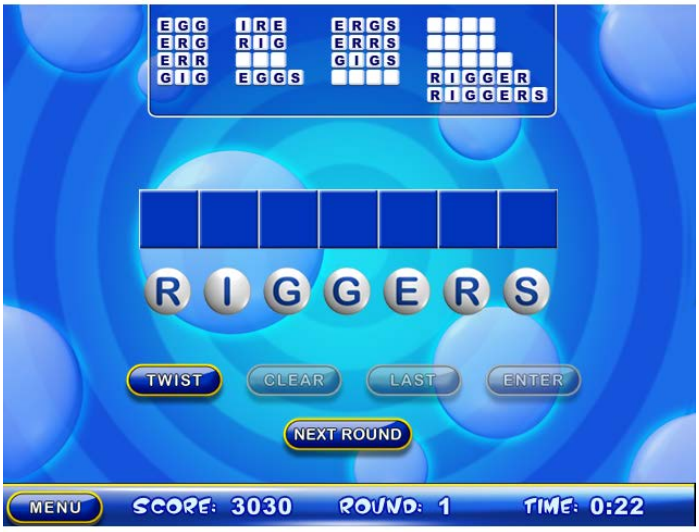
CS2201
Fall 2019
Assignment No. 10

Purpose: This project is meant to give you experience with a recursive, backtracking solution algorithm.

Assignment: Your task will be to design and implement an algorithm to find words in a Text Twist game.

The game of Text Twist: In the game of Text Twist, you are given seven letters from which you are to form as many words as possible that are three or more letters long. You are given two minutes in which to do your work. Find at least one word that uses all the letters to continue playing to the next round. Bonus points are awarded if you find all of the words. To play, you click on the balls or type in the letters to form a word, then click the ENTER button or press the ENTER key. If the word is in the game's dictionary, it will show up on the top of the game. Use the TWIST button when you get stuck to help you see other words.



Your assignment is to write a program that finds all words, of three or more letters, that can be generated from the seven given letters. You will use the Trie class you developed in the previous lab to keep a lexicon of acceptable words, as well as to keep track of the words you have found.

Requirements:

1. You are to submit all of your **source code** files (.h & .cpp files)**.** The files to be submitted include: `Trie.h`, `Trie.cpp`, (with `TrieNode.h` and `TrieNode.cpp` if appropriate) and `TextTwist.cpp` (and accompanying .h file if appropriate). The functional specification for the Trie (and TrieNode) class was specified in the previous lab. You should fix all errors in your Trie class that were identified during grading before proceeding with the rest of the lab. The `TextTwist.cpp` file will contain your main program, and associated functions, which finds words in the Text Twist game. If you want, you may design a class to represent the game, and then write your main program in another file (say `texttwistdriver.cpp`). Be sure to submit all the files required to create an executable program. Please also submit your CMakeLists.txt file. Do not submit ospd.txt.
2. All functions should be properly documented with pre- and post-conditions. These comments should appear both with the function prototype and the function definition.
3. In addition to your source code files, you are to submit a **README** document (a .txt text file or a .doc Word document) that answers the questions listed later in this document.

Functional Specifications:

You will begin the game by building a lexicon of acceptable words. The lexicon will be loaded from the file `"ospd.txt"`, which contains the words from the Official Scrabble Players Dictionary (you can find additional word lists here). Then you will prompt the user to enter the seven letters of the Text Twist game and read in those characters from the terminal. After reading in the word and verifying that it is seven characters long and only contains lowercase

letters, your task is to find all possible words by generating all possible combinations of those letters with a recursive backtracking algorithm and determining which combinations are actual words. Store all words found in another lexicon. After your search is completed, print all words found in lexicographical order (simply using the lexicon's `print` method). You are not required to ask the user if they want to run the program again (you can if you want, but it adds an extra level of complexity).

When searching for words, your program must check the following conditions for each potential word:

- that the word is at least three letters long
- that the word is defined in the lexicon as a legal word
- each letter of the game is used at most one time (you cannot use a letter twice unless it is given twice)
- the current potential word has not been previously created and considered (can occur if there are multiple instances of a letter)

As with any exponential search algorithm, it is important to limit the search as much as you can to ensure that the process can be computed in a reasonable time. One of the most important strategies is to recognize when you are going down a dead end so you can abandon it. For example, if you have built a word starting with "zx", you can use the lexicon's `isPrefix` function to determine that there are no words in the English language beginning with that prefix. Thus, you can stop right there and move on to more promising combinations. If you generate all possible permutations of the letters you will not receive full credit on this assignment, and you'll find yourself taking long coffee breaks while the computer is busy checking out non-existent words like zxgub, zxaep, etc. Similarly, you will want to make sure that you do not explore words that have previously been explored. This can happen if a letter is specified twice in the list of seven letters. Consider the figure above where there are two R's and two G's. Once we have found words that start with "RIG" we do not want to reconsider those same words when we try letter combinations that use the second R in place of the first R or the second G in place of the first G.

Note: I am giving you a great deal of flexibility in how you design your solution for this lab. Put a lot of thought into your design ***before*** you put much effort into your coding. Take my word for it: a good design will make this programming assignment fun and enjoyable, while a bad design will be a nightmare. If you seek help from either the instructor or the TAs, our first question will be "What is your design for this program?" Keep things simple, but make sure you have thought about all the issues. Note: besides grading your solution for correctness, we will also be grading your design. There are four criteria on which your design will be graded: (1) modularity, (2) encapsulation and information hiding, (3) the design of your recursive backtracking solver function, and (4) overall program structure. These criteria will constitute **20%** of your grade on this assignment. Each file you turn in should have the standard block comments at the top of the file, including an honor statement. Be sure to use good programming style.

When creating your project, make sure your CMakeLists.txt file contains all the compiler flags that we have been using on all assignments this semester. The compiler flags that should be included in your CMakeLists.txt file are:

```
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
set(CMAKE_CXX_EXTENSIONS OFF)
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++17 -Wall -Werror -Wextra -pedantic -pedantic-errors")
set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${CMAKE_CURRENT_SOURCE_DIR})
```

Also make sure that you have CLion set up to use the clang compiler. See the supplemental document that was distributed with project #6 that shows you how tell CLion to use the clang compiler.

When you want to create new files and have them added to the project, right-click on the CMakeLists.txt file in the project pane and select `New=>C++ Class`. Specify the name of the class (it will create both .h & .cpp files for you), and make sure the "**Add to targets**" option is selected. This last step will ensure that the files get added to the CMakeLists.txt file correctly so that they get compiled.

Special Instructions:

A set of files associated with this project are available from the project page in Brightspace. The following files are available:

- `ospd.txt` -- the official Scrabble player's dictionary (it contains 79,339 words).
- `texttwist_Win.exe` -- a sample executable for Windows
- `texttwist_MacOS` -- a sample executable for MacOS

README: Answer the following questions in your `README` document (a .txt text file or a .doc Word document).

1. List your name and email address.
2. After reviewing this spec and the provided files, please estimate/report how long you think it will take you to complete it.
3. How many hours did you actually spend total on this assignment?
4. Did you access any other reference material other than those provided by the instructor and the class text? Please list them.
5. What did you enjoy about this assignment? What did you dislike? What could we have done better?
6. Was this project description lacking in any manner? Please be specific.

Submission:

You are to submit all of your **source code** files (.h & .cpp files) that are necessary to build and run your program**.** Please also submit your **CMakeLists.txt file**. The files to be submitted include: `Trie.h`, `Trie.cpp`, (with `TrieNode.h` and `TrieNode.cpp` if appropriate) and `TextTwist.cpp` (and accompanying .h file if appropriate). The `TextTwist.cpp` file will contain your main program, and associated functions, which finds words in the Text Twist game. If you want, you may design a class to represent the game, and then write your main program in another file (say `texttwistdriver.cpp`). Be sure to submit all the files required to build & create an executable program. Do not submit ospd.txt. Again, remember to submit your CMakeLists.txt as well as your README file.