CS-231 Haskell Language Project 3

Winter 2022

Overview

You will write a program to play a game based loosely on 2 games, "wordle" and "Fermi, Pico, Nano". In wordle, you are given a word of 5 letters to guess, all your guesses are words in the dictionary, and you are shown the results of your guess, which is a coloring of the letters to indicate a letter in the correct position (green), a letter in the word but in the wrong position (yellow), or a letter not in the word (black). In Fermi, Pico, Nano, there are 2 players. The first player chooses a sequence of digits; the second player guesses by communicating a sequence of digits, and the first player gives a response consisting of Fermi(s), Nano(s), and Pico(s), always in alphabetic order where Fermi means there was a digit which was correct and in the correct position; Nano means that there is a digit which does not occur, and Pico means there is a digit which is in the sequence, but is in the wrong place. Thus if player 2 guesses correctly, the result is Fermi Fermi Fermi. If player 2 guesses 3 digits which are not in the sequence, the result is Nano Nano Nano, If player 2 guesses exactly 2 correct digits, but they are in the wrong positions, the result is Pico Pico Nano.

Specifics

In your program, you will choose a 5 letter word which is stored in a file (details supplied later). Your program will ask the user to guess a word (which must be in the dictionary). The user will be given a 5 digit number which will tell him how good the guess was. The number will be created in this fashion:

- For every correct letter in the correct position, the digit 3 will be in the number. All digits 3 which occur will be in the leftmost positions of the number.
- For every correct letter in the wrong position, the digit 7 will be in the number. Any and all 3s will proceed the first 7 (if any) in the number.
- For every incorrect letter, the digit 9 will appear in the number. All digits 9 which appear will be in the rightmost positions of the number.

For example, if the word is "digit" and the guess is "octal", the response number will be 99999. If the word is "digit" and the guess is "dodge", the response number will be 37999. If the word is "digit" and the guess is "digit", the response number is 33333. Of course, a further response will state that the player has won.

The program will play one game and stop. To guess a second word, run the program again.

When you run the program, you will have as a command line argument the name of a dictionary file to check user guesses against. This file will not be altered in any way by your program.

A user will have a maximum of 8 guesses in a play of the game. A guess is a 5 letter word which is found in the dictionary. If the user enters a guess which is not a word in the dictionary, that guess will be rejected (will not count against the 8 possible guesses the user is allowed).

Selecting the Word to Guess

Your program will check whether the file "tmpDictionary.tmp" exists. If it does not, your program will select the word from the 5 letter words in the dictionary named in the command line argument. Before your code exits, the 5 letter words, except the one selected for play, will be written to file "tmpDictionary.tmp". (The quotes are not part of the file name, they are there to emphasize that file names are Strings in Haskell).

If the file "tmpDictionary.tmp" does exist, the word to guess will be selected from its contents, and before your code exits, that file will be replaced with a file of the same name which contains all words that were read from the file except the word selected for the game.

Your program will also check whether the file "tmpPrime.tmp" exists. If it does, it will contain one number. Your program will read that number, say n, then compute the nth prime, say p. Your program will take the element of the dictionary at position p mod dictionary size (in case p is larger than the size of the dictionary). Your program will replace "tmpPrime.tmp" with a file containing the number generated by (n * (prime 1101) `mod` 1103)+7.

If "tmpPrime.tmp" does not exist, have the user enter a number between 1 and 10. Add that to 1100, then use that value as n in the discussion above. In this case, you will be writing "tmpPrime.tmp" without replacing it, because it does not exist yet - but it will the next time the game is played.

Input

The input comes from the dictionary, whose name is a command line argument, the files "tmpDictionary.tmp" and "tmpPrime.tmp" if those files exist, and from the user. The user will type words for his guesses at the keyboard.

Processing

Your program will input the dictionary to check user input words, and will also select the word for play in the manner described above. The user will be presented with an introduction including the name of the game and an option to read instructions about how to play. The program will play up to 8 rounds of guessing, quitting sooner if the user guesses correctly sooner than guess 8. Each guess from the user will be checked against the dictionary. Only actual words will be allowed for guesses, other "word" inputs will be rejected without being counted as user guesses. For every guess, a 5 digit number will be generated and shown to the user. If the user finishes guess 8 without guessing the word, a message that the user has lost will be displayed. If the number 33333 is generated, a message that the user has won will be displayed.

Output

All text displayed on the screen will be meaningful and easy to read and understand. The player instructions will be concise, accurate, and understandable. Note that the instructions are only displayed if requested by the player.

Notes

1. Your use of the Haskell do keyword will be limited. The only parts of the code where do is allowed is in functions which perform input and/or output.
2. Much of this program is input/output.
3. You will need to determine which modules to import in order to do such things as delete or rename files.
4. Use the dictionary file from project 2 for this project as well.

Grading

This is an 80 point program. Write all your code in one file. You must turn in a program which compiles in order to receive any credit for this assignment.

Deliverables

Your source code file must be submitted via Blackboard no later than 11:59 p.m. Friday March 25.