CS 496: Assignment 1
Due: May 31, 11:55pm

1 Assignment Policies

Collaboration Policy. It is acceptable for students to collaborate in understanding the material but not in solving the problems or programming. Use of the Internet is allowed, but should not include searching for existing solutions.

Under absolutely no circumstances code can be exchanged between students. Excerpts of code presented in class can be used.

Assignments from previous offerings of the course must not be re-used. Violations will be penalized appropriately.

2 Assignment

This project seeks to help you get some practice with lists and tuples. It is based off of the Wordle game from New York Times (`https://www.nytimes.com/games/wordle/index.html`). If you haven't played this game, what are you doing?! This implementation looks a lot simpler compared to the original. The UI is composed of ASCII characters.

To run the game, first install an `opam` package `csv`. This is needed to load a word bank of 5 letter words. You are welcome to add additional words to the word bank. Then, in `utop`, run

```
#use "wordle.ml";;
```

You can quit the game by entering `quit` or `exit`.

2.1 Game State of ASCII Wordle

The game state has three parts: `w_h, kb, the_word`.

The first part is wordle history. It keeps track of all the guesses made. Since there are a total of 6 guesses of 5 letter words, it is a list of 6 strings, initially comprised of 5 underlines each. The underlines are also used for display purposes. Every time a word is guessed, a blank string is replaced by the guess, starting from the left. Wordle history is named `w_h` as shown below:

```
let w_h = ["_____";"_____";"_____";"_____";"_____";"_____"]
```

The second part is keyboard. It is a list of strings that mimics the keyboard layout. The spaces are there for display purposes. As the game progresses, letters that are not present in the target word will be removed from it. Keyboard is named `kb` as shown below:

```
let kb = ["QWERTYUIOP";" ASDFGHJKL";"  ZXCVBNM"]
```

The third part is the target word. It is a hard coded answer converted to uppercase. You can change the target word to any 5 letter word, as long as it is in the word bank. The game also converts all player inputs to uppercase. The target word is named `the_word` as shown below:

```
let the_word = String.uppercase_ascii "ocaml"
```

2.2 Exercises

1. Implement a function

   `process_data : string list list -> string list`

   that converts the word bank `csv` data `word_bank_src` that is a list of lists of strings in to a list of strings. You must also convert every string to uppercase.

   Propose two solutions:

   (a) `process_data` without using fold (e.g. using explicit matching `match`)

   (b) `process_data_fold` using fold.

2. Implement a function

   `is_word_valid : string -> bool`

   that checks whether a word exists in the word bank `word_bank`.

3. Implement a function

   `space_out_word : string -> string`

   that returns a new string where characters are separated by one space character. For example

   ```
   # space_out_word "OCAML";;
   - : string = "O C A M L";;
   ```

   Trailing space is allowed.

4. Implement a function

   `rm_letter : string list -> char -> string list`

   that given the current keyboard and a letter to remove, returns a new keyboard where the letter is replaced by a space character. For example:

```
# rm_letter kb 'D';;
- : string list = ["QWERTYUIOP"; " AS FGHJKL"; "  ZXCVBNM"]
```

You must use recursion for this function.

5. Implement a function

   `new_kb : string list -> string -> int -> string list`

   such that `new_kb kb word idx` returns a new keyboard with letters from `word` that are not present in `the_word` removed from `kb`. `idx` is used track the nth character in `word`, starting from 0. For example with `the_word = "OCAML"`:

   ```
   new_kb kb "WRONG" 0;;
   - : string list = ["Q E TYUIOP"; " ASDF HJKL"; "  ZXCVB M"]
   ```

   hint: use `rm_letter`

6. Implement a function

   `fill_in_the_blanks : string -> string -> string`

   such that `fill_in_the_blanks blanks word` returns a string that tells the player which of the 5 letters have been guessed. Due to limitations of ASCII display, we will communicate the progress to the player via `letters in the right spots` and `correct ↪ letters`.

   `letters in the right spots` is 5 blanks that will be revealed once the correct letter at the correct spot has been guessed. For example with target word `"OCAML"`, if we have guessed `"OCCUR"`, then `letters in the right spots` would be `OC___`.

   `correct letters` is a list of letters that are letters in the target word, but are not one of `letters in the right spots`. For example with target word `"OCAML"`, if we have guessed `"MASON"`, then `correct letters` would be `O A M`.

   For this function, we only focus on `letters in the right spots`. For example with target word `"OCAML"`, if we guess the word `"CAMEL"` and the first letter `"O"` has already been revealed:

   ```
   # fill_in_the_blanks "O____" "CAMEL";;
   - : string = "O___L"
   ```

7. Implement a function

   `rm_dup : 'a list -> 'a list`

   that removed duplicates from a list. You must NOT use recursion. For example

   ```
   # rm_dup ["M";"C";"D";"C"];;
   - : string list = ["M"; "D"; "C"]
   ```

8. Implement a function

`get_good_letters : string -> string list`

such that `get_good_letters word` returns get a list of the letters from `word` that are also in the target word `the_word`, regardless of position. For example with target word `"OCAML"`:

```
# get_good_letters "MILLS";;
- : string list = ["L"; "L"; "M"]
```

9. Implement a function

`pcl_helper : string -> string list -> string`

that helps function `print_correct_letters` format the `correct letters` section, as mentioned in exercise 6. The function `pcl_helper blanks good_letters` returns a string composed of spaced out letters that are present in the target word `the_word`, but NOT present in the string `blanks`. For example with target word `"OCAML"`:

```
# pcl_helper "OCA__" ["O";"C";"M";"L";"L"];;
- : string = "M L"
```

3 Submission instructions

Submit a single file named `wordle.ml` through Canvas. No report is required. Your grade will be determined as follows:

- You will get 0 points if your code does not compile.
- Partial credit may be given for style, comments and readability.