Learning to Program with F#
Exercises
Department of Computer Science
University of Copenhagen

Jon Sporring, Martin Elsman, Torben Mogensen, Christina Lioma

October 21, 2022

0.1 Random text

0.1.1 Teacher's guide

Emne Functional programming, histograms, random values, tree types

Sværhedsgrad Hard

0.1.2 Introduction

H.C. Andersen (1805-1875) is a Danish author who wrote plays, travelogues, novels, poems, but perhaps is best known for his fairy tales. An example is Little Claus and Big Claus (Danish: Lille Claus og store Claus), which is a tale about a poor farmer, who outsmarts a rich farmer. A translation can be found here: `http://andersen.sdu.dk/vaerk/hersholt/LittleClausAndBigClaus_e.html`. It starts like this:

> "LITTLE CLAUS AND BIG CLAUS a translation of Hans Christian Andersen's 'Lille Claus og Store Claus' by Jean Hersholt. In a village there lived two men who had the selfsame name. Both were named Claus. But one of them owned four horses, and the other owned only one horse; so to distinguish between them people called the man who had four horses Big Claus, and the man who had only one horse Little Claus. Now I'll tell you what happened to these two, for this is a true story".

The assignment is to design a spell checker that is based on the text by H.C. Andersen. As part of this handout please find `spellCheck.fsx`, which you should complete by solving the exercises of this assignment.

0.1.3 Exercise(s)

0.1.3.1:

(a) Please read and understand the data structure Trie that has been handed out in `spellCheck.fsx` and understand how a word tree is created in this Trie and how the operations are performed.

(b) We need to create an `autoComplete` functionality based on a lookup of a specific word in the trie. This function `lookup` should have the signature `let lookup (prefix: string) (trie: Trie<char>) : Trie<char> Option`. It should return an option of the (sub)trie found by the lookup. `autoComplete` will use `lookup` to check if the word beginning with the `prefix` exists, after which it should return a sequence of strings of words which `prefix` in the trie. The signature of `autoComplete` is `let autoComplete (prefix: string) (trie: Trie<char>) : string seq`.

(c) Implement `spellCheck` with the signature `let spellCheck (word: string) (trie: Trie<char>) : bool` with the functionality to check if a given word is contained in the trie.

(d) Implement `genText` with the signature `let genText (len: int) (trie: Trie<char>) : string` which generates a string text of length `len` of random words that are generated by the function `randWord` with the signature `let randWord (trie: Trie<char>) : char list`. `randWord` should retrieve a random word from the given trie.

(e) Test your implementation with the given tests and, potentially, add extra tests.