Assignment 1
T9
v1.2.1
Programming Languages

The deadline for this assignment can be found at: TimeEdit

• The submission should consist of:
  – The complete report, including
    * A front page with the following information:
      1. Your name.
      2. The course name.
      4. The assignment number.
      5. The version of the submission (in case of re-submissions).
  – An appendix with the source code.
  – To simplify feedback, the main report (excluding the appendix) must have numbered sections and page numbers.

• If you write your report using LATEX, double-check that your references are ok and not e.g., “Figure ??” before you submit.

• Submit your report as a pdf file uploaded via labres.

• Furthermore, the source code should be available in a folder called edu/5dv086/ass1 in your home folder. You will probably have to create the folder yourself. Ensure that the folder is world-non-readable and group-readable (chmod 755 ~/edu/5dv086 ~/edu/5dv086/ass1 should do it). Further instructions on the source-code follows in section 5.
1 Introduction

"Haskell is one of the leading languages for teaching functional programming, enabling students to write simpler and cleaner code, and to learn how to structure and reason about programs."

Graham Hutton, *Programming in Haskell*

This assignment consists of three parts: Theory, algorithms, and applications. The purpose of the assignment is primarily to demonstrate your theoretical knowledge, your ability to express yourself clearly in writing, your ability to construct, understand and implement algorithms declaratively and recursively as opposed to iteratively, and showcase your ability to solve practical problems using the functional programming paradigm.

The assignment shall be performed individually. You are allowed (indeed encouraged) to discuss the assignment with other students. However, all solutions, theory, code, etc. should be constructed by yourself. You may be required to explain your submitted assignment.

Your report should clearly explain all important aspects of your solution. These include, but are not limited to, the design of internal data representations and explanations of complex algorithms and data flows. The primary purpose of the report is to practice technical communication. The algorithm description is of particular importance. The report should be written in English.

2 Background

An SMS, or short messaging service, is a type of communication used in mobile phones. We refer to these messages as “text messages”. With SMS, mobile users can communicate via one another asynchronously.

Preceding the smartphone era most cell phones had a numerical keyboard on which to enter text. Inputting messages using only ten keys posed a problem. A standard solution to this problem was to assign multiple letters to each digit. To enter a particular letter the user is supposed to press its corresponding digit repeatedly until the correct letter appears on the screen.

In this assignment, we consider the layout of letters shown in Figure 1. The 0-digit represents a blank.

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ABC</td>
<td>DEF</td>
</tr>
<tr>
<td>4</td>
<td>GHI</td>
<td>JKL</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>MNO</td>
</tr>
<tr>
<td>7</td>
<td>PQRS</td>
<td>TUV</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>WXYZ</td>
</tr>
<tr>
<td>0</td>
<td>Space</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Standard keyboard mapping

\[1\] If you find the distinction difficult, use this rule of thumb: If you have constructed everything yourself and understand and can explain what you have done and why you have done it, you should be OK.
Dictionary-based systems such as T9 reduce the average number of keypresses needed to input a message. They work by the principle that a single keypress represents one of the letters associated with that key. For example, the sequence 738 might refer to the words "PET" or "SET". To disambiguate between several possible words, we will choose the word that, according to the dictionary, is most common. Ties are broken by selecting the first word in lexicographical order.

3 T9

In this assignment, you are required to implement a function in Haskell that returns a minimum sequence of keypresses that produces a given message for a particular dictionary.\(^2\) The definition for “message” and “dictionary” is given in subsection 3.1.

3.1 Terminology

We define the following set of rules and definitions,

- A **message** is a string of uppercase letters A-Z and blanks (spaces).
- An **input string** (i.e. a T9 encoding of a message) is a sequence of digits (0, 2-9) and the special character "^".
- A **prefix** is a sequence of digits /"^[2-9]+\$/(regex) representing letters.
- The character "^" appended to a prefix modifies that prefix and yields another prefix.
- A prefix can be modified any number of times.
- The digit “0” maps into a blank (space).

**Definition 3.1.** Dictionary A **dictionary** is a set of tuples of the type (String, Integer) where the String is a word and the Integer expresses the frequency with which the word is likely to occur. For an example, \(d\) in **Listing 1** is an unsorted dictionary.

```
\(d = [("FOO", 3),
("BAR", 4),
("BAZ", 4)]\)
```

**Listing 1:** An unsorted dictionary

3.2 Sorting a dictionary

To sort the dictionary, we first order the dictionary entries in decreasing order of frequency. If two or more entries have the same frequency, they are ordered lexicographically.

Hence, the sorted version of \(d\) from **Listing 1** is shown in **Listing 2**

\(^2\)There may be multiple T9 encodings of the same string that is of the same length
Both 'BAR' and 'BAZ' occur with the same frequency and must be compared lexicographically. Since 'BAR' satisfies the relation

'BAR' < 'BAZ'

it becomes the "first" entry.

3.3 Translating a prefix to a word

In this section, we describe how to translate a prefix to a word. It is from this portion of the assignment specification that you will be able to derive how to express any given word using the shortest possible T9 encoding.

This section is intentionally informal with respect to the algorithm description and we provide only a small and limited example.

We say that the prefix $p$ together with the dictionary $d$ determines the set of possible words that $p$ might represent and it is by performing the following sequence of operations that we may translate $p$ to a particular word $w$ in the dictionary,

1. Begin by sorting the dictionary entries in decreasing order of frequency.
2. Any ties are resolved by comparing the words in the entries lexicographically.
3. If there are no possible words, then the prefix maps into the special character "?".
4. A prefix which has been modified $N$ times maps into the $(N + 1)$-th word in the dictionary.
5. If $(N + 1)$ exceeds the length of the dictionary then it wraps around the dictionary.

3.3.1 Example

Consider the already sorted dictionary $d$,

$$d = \{ ("FCQ", 3), ("DBR", 2), ("TOF", 1) \};$$

and the input string $p=32^0804^{^\sim\sim}$. We have that $p$ consists of three prefixes, namely $32^-$, $8$, and $4^{^\sim\sim}$. The sequence $32^-$, together with the dictionary, maps to two distinct possibilities in the order '"FCQ' and '"DBR'. Since the sequence is modified once by the character "\~\~", the word corresponding to $32^-$ is '"DBR'. The second prefix $8$ maps into only one possibility, namely '"TOF'.

\textsuperscript{3}Duplicates entries are not allowed within a dictionary
The third prefix, $4^{\infty}$, does not match any of the words in the dictionary and therefore it maps to “?”.

Hence, the message corresponding to the input string $32^0804^{\infty}$ is 'DBR TOF ?'.

You have to solve the slightly more interesting inverse problem of finding a minimum sequence of keypresses which generates a given message.

We restrict ourselves to messages which can actually be entered with the given dictionary. For example, the message "DBR TOF" is minimally entered with the input string $3^08$.

Specifically, your function should be named $t9$ and should be of the type,

```
Listing 3: The type signature of $t9$
```

where the first argument is a dictionary, the second argument is the message, and the result is an associated (shortest) input string.

4 Example(s)

The easiest way to exercise your implementation is against examples which you can verify by hand. Below we present but a few of these examples,

**Example 4.1.** For both the dictionary $d$ and $d'$ where

$$d = \{("BANANA", 7), ("ABBA", 4)\}$$
$$d' = \{("BANANA", 7), ("A", 4)\}$$

the minimum input string for 'A' and 'ABBA' is $2^\infty$.

**Example 4.2.** For the dictionary

$$d = \{("FOOP", 2),
\{("FOOQ", 2),
\{("FOOR", 2),
\{("FOOS", 2)\}\}$$

the minimum input string for 'FOOS' is $3^{\infty}$.

**Example 4.3.** For the dictionary

$$d = \{("NO", 2),
\{("NOW", 2),
\{("NOWAY", 2)\}\}$$

the minimum input string for 'NOWAY' is $5^{\infty}$.

---

4Do not forget that you can write these out as tests in some file or use a testing library
4.1 More exhaustive examples

In these sets of examples we show some of the possible (minimum) encodings — there may be several — that all use the dictionary in Dictionary.hs for some of the messages that are found in Messages.hs.

Example 4.4. For the message,

THOSE WHO DO NOT HAVE GOALS ARE DOOMED TO WORK FOR THOSE WHO DO

one of the minimum encodings is

\[ 846^094^03060^4^0460^2^03660^809670^3^0846^094^03 \]

Example 4.5. For the message,

THE SHORTEST WAY TO DO MANY THINGS IS TO DO A SINGLE THING AT A TIME

we have that both

\[ 8^07409208030^62^608440408030^2074^0844^0280208^-^-^-^0844^0280208^-^-^- \]

and

\[ 8^07409208030^62^608440408030^2074^6^0844^0280208^-^-^- \]

are valid input strings.

5 Submission Rules

As mentioned earlier your source code should be available in a folder called edu/5dv086/ass1 in your home folder. You are free to distribute your solution across multiple files, however one file has to be called T9.hs and it is only allowed to export the t9 function, i.e. your file must start with the following

```
module T9 (
    t9,
) where
```

Listing 4: Obligatory export of the t9 function

this is to afford us the ability to automatically test your assignment.

- Late submission will receive zero points and will be graded together with the first batch of resubmissions.
- It is not allowed to copy the work of others. The submission must be the original work of the individual.
- The grader reserves the right to interview the individual about the solution.
- No paper submission is required. Please do not submit paper versions.

6 Test script

For your convinience we have supplied you with a non-exhaustive test-script avaible here Main.hs. Save it in the same directory as your T9.hs file, and compile it using `ghc`. The script will only produce output
when there is an error or a possible error. Successful tests do not produce any output.

7 Further notes

7.1 Type Synonyms

You are allowed to use type synonyms,

```
type Frequency = Integer
type Word = String
type T9Encoding = String
type Message = String
type Dictionary = [(Word, Frequency)]
t9 :: Dictionary -> Message -> T9Encoding
```

Listing 5: The type signature of t9 using type-aliases

7.2 Functors, Applicatives, and Monads

It is not necessary to use Functors, Applicatives, and/or Monads to solve this assignment.

7.3 Error handling

For this assignment, there are no requirements on the robustness of your program although you are very much encouraged to use “safe” operations.\(^5\) The robustness of your solution will be reflected in your score on this assignment.

8 Suggested Reading and Additional Resources

We humbly suggest that you read Learn You A Haskell For Great Good! to get started with Haskell. It is generally a well perceived text that we have used in the past on this course.

Beyond that some students have had an easier time learning Haskell by doing small practice problems at Code Wars. One beneficial aspect of Code Wars is that after completing a problem one gets to see other solutions to that same problem.

There are also some old assignments from this course that are still available online.

Additional resources include, in no particular order, the following texts:

- Setting up a Haskell project
- What I Wish I Knew When Learning Haskell
- Real World Haskell

\(^5\)However, if your program crashes on valid input it is not robust enough.
• What to read after reading Learn You A Haskell and Real World Haskell??
• You Could Have Invented Monads!
• A Haskell Reading List