German University in Cairo
Media Engineering and Technology
Prof. Dr. Slim Abdennadher

CSEN403: Concepts of Programming Languages, Spring 2015
Haskell Project

Submission: Tuesday 21.04.2015 (11:59 pm)

The project will put your knowledge in Haskell to the test. Before proceeding, make sure that you read each section carefully. Enjoy.

Instructions

Please read and follow the instructions carefully:

a) The project should be implemented using Haskell (Hugs98) based on the syntax discussed in class.

b) This is a team project. You can work in groups of **maximum** 3 members. The groups for this project are the same as project 1. In case of any changes, please report it by sending an email to your TA maha.samir@guc.edu.eg or ahmed.el-sawy@guc.edu.eg. You must send complete information about your new team including full names and unique GUC application numbers. The deadline for sending group change requests is Wednesday 08.04.2015 (11:59 pm).

c) You have to write a **formal report** on your project covering the following points:

- A brief formal description of the datatypes that you have created.
- A brief formal description of the *main* functions used in your implementation.
- A listing of the *helper* functions that you called from your main functions. You are allowed to use built-in Haskell functions, like `length` or other functions in `Hugs.Prelude` without explanation: `https://www.haskell.org/onlinereport/prelude-index.html`

d) Every team has to submit a soft copy of the following files (in a zipped directory) to the following email address csen403s15@gmail.com:

- your Haskell project **source file** named after your team code, e.g. `T10G03.hs`
- the report file (.doc or .pdf) also named after your team code , e.g. `T10G03.pdf`
- the zipped file should also be named after your team code, e.g. `T10G03.zip`
- your email subject has to be formatted as follows: Project␣2␣TeamCode, e.g. Project 2 T10G03
- it is **your** responsibility to ensure that you have submitted the correct files and saved the correct content before attaching.

e) Every team has to submit a **proper** *hard copy* of the following:

- the report
- a printout of the source file

f) You are always welcome to discuss the project with the TAs. You must work with your team members only. Do not exchange information with other teams or individuals.

g) Cheating and plagiarism will be strictly punished with a grade of 0 in the project.

h) Once you submit the project, you will be appointed a date to show up with your team members for an **oral evaluation** of your project. The evaluation will cover practical and technical details as well as theoretical concepts concerning Haskell and the general features of the functional programming paradigm.

i) Please respect the **submission deadline** marked at the beginning of the document as well as the date of the oral evaluation set by your TA. Any delay will result in a rejection of the submission and a cancellation of the oral evaluation.

Project Description

You are required to implement a simple algorithmic music composer. It should be able to learn some statistics from a number of given strings representing famous musical pieces and then use these statistics to generate new strings representing new pieces.

You will be using this website `http://virtualpiano.net/`. It provides a virtual piano on which you can play using the keyboard. Moreover, it provides music sheets of a collection of famous songs.

For this project, we have chosen a number of strings from those provided on the virtual piano website and modified them a bit so that they would include only the characters `[0-9]`, `[a-z]` and `['B','C','D','E','G','H','I','J','L','O','P','Q','S','T','V','W','Y','Z']`. The file `MusicResources.hs` will contain these strings. Your code should use the list `training` from that file in order to learn. Make sure you have that file in the same folder as your solution .hs file. Then, you can write in your file:

```
import MusicResources
```

In order to implement the composer, you are required to:

a) Implement the function **makeStatsList** which will **dynamically** generate a list of statistics according to the content of the `training` list in the MusicResources.hs file. The type of `makeStatsList` will be:

```
makeStatsList :: [(Char,[(Int,Char)])]
```

The size of the output list will be 54 items since we have 54 different characters to gather statistics about. Every entry in that list is a pair representing statistical information about a specific character which is the first item of the pair. The second item in the pair is a list of pairs containing which other characters appeared directly after that specific character and with what frequency. Each inner list has to be **sorted** from the highest to the lowest frequency value.

b) Implement the function **compose**:

```
compose :: Char -> Int -> [Char]
```

The input to this function will be a character representing the start of the musical piece and an integer representing the total length of the musical piece that should be generated. The `compose` function has to call `makeStatsList` to use the information gathered in the composition process as follows: For every character in the output string, the next character will be determined as a random variable that can take values from the set of next possible characters according to our statistics. However, when the value of the next character is being randomly chosen, the weights of the frequency of each option have to be taken into consideration. For example, assume the currently character is `'O'` and the statistics pair of `'O'` is:

```
('O',[(8,'w'),(2,'q')])
```

Then, the next character chosen should be 'w' with a probability of 80% and 'q' with a probability of 20%. This is because the only two possible next characters in the training strings are 'w' and 'q' with frequency 8 and 2, respectively. Hint: You can use the function `randomZeroToX` in your solution which you will find in `MusicResources.hs`.

Sample Test

a) The file `makeStatsListCall.txt` shows the output of a sample `makeStatsListCall` that was generated with a value of the `training` list equal to:

```
training =  [furElise, moonLightSonata,secondWaltz, preludeInCMajor, caspersLullaby]
```

b) The fun part of this project is that you can open the virtual piano website and play the strings your code outputs in part b of the project. We are providing you with a script file that will help automate the process of sending keystrokes corresponding to the musical piece string to Mozilla Firefox.

Example:

```
Main> compose 'e' 25
"euoyuoetusauotyuodgdadgti"
```

Grading and Evaluation

The evaluation of the practical part of the project, i.e the implementation, is the same for the whole team. However, the oral evaluation is team-member-specific. Your oral evaluation will affect your overall project grade. So be prepared :)

Tips

Here is a list of the do's and don't for a better performance in the project as well as in the course:

- Read carefully the lecture notes and revise the slides and practice assignment problems. They include a lot of useful information for you. Moreover, check the links on the course page for further help: `http://met.guc.edu.eg/Courses/Links.aspx?crsEdId=561`
- Start working early on the project. Do not get carried away and keep track of time. It would be handy if you planed ahead your project and worked out a reasonable timeline.
- Divide the work among your team members. Coordinate your efforts together and meet regularly to update each other with the progress.

Finally, we wish you the best of luck!