Haskell Programming Assignment

Functional Programming

April 11, 2019

Contents

1 Introduction

Your Functional Programming is made up of 50% Final Exam and 50% Continuous Assessment. The continuous assessment is made up of

- In-class lab test (Week 9) - 5% (more details here)
- Programming Assignment (due week 13) - 40%
- In-class lab test - 5% As this test has been cancelled, you will be give the best if your other two marks for this 5% slot.

2 Single Transferable Vote

The single transferable vote (STV) is a voting system designed to achieve proportional representation through ranked voting in multi-seat constituencies. A good video here. You can look up lots on STV but from the point of view of the assignment, we are interested in the counting of the votes. Specifically the rules for our election are:

1. All valid papers are grouped by first preference votes and candidates are ordered in descending order of first preferences. Each vote is given a weight of 1000.

2. The quota is calculated

$$quota = (\frac{number\ of\ valid\ votes}{number\ of\ seats\ +\ 1})\ +\ 1$$

3. If, at the end of any count, a candidate has a total weight of votes greater than the quota, then this candidate should be elected.

4. Calculate the total weight of transferable votes. If the total weight of transferable votes is greater than the surplus, then transfer each of the votes (to the continuing candidate indicated as the next available preference) with a reduced weight,

$$newweight = old\ weight * (\frac{surplus}{total\ weight\ of\ transferable\ papers})$$

   If the total weigh of transferable votes is less than or equal to the surplus, transfer all votes in the bundle, leaving the weights unchanged.

5. If, at the end of any count, no candidate has reached the quota and there are more continuing candidates than vacancies, a candidate must be eliminated. Working backwards from the candidates with the lowest weight, distribute the (eliminated) candidate's votes, leaving the weights unchanged.

6. **STOP** when the number of elected candidates + then number of candidates remaining (uneliminated) = the number of seats. At this point the remaining (unelected and uneliminated) candidates are deemed to be elected 'without reaching the quota'.

3 Assignment Details

You are asked to write a Haskell program to count votes in an election using Single Transferable Vote.

- The output should contain **at least** the list of elected candidates in order of election.
- You should use the voting example from Hutton (as published here) to start.
- Specifically, you can use the type of input as follows:

```
ballots :: [[String]]
ballots = [["Red","Green"],      -- i.e. "Red" is first preference,
                                 -- "Green" is second preference.
           ["Blue"],
           ["Green","Red","Blue"],
           ["Blue","Green","Red"],
           ["Green"]]
```

   (For extra marks, you can import from file into this structure)

4 Marking Scheme

Functionality	%
Elected Candidates Corrrect	70
Extra Information	
Count number with elected candidates	10
Input coming from file	7
More comprehensive information per count	8
Miscellaneous extras (aka very impressive stuff!)	5

note changed marks for functionality

Quality of Haskell Code	%
Haskell Style	50
Elegance including naming	30
Use of appropriate libraries, structures	20

The final weighting between functionality and quality of Haskell Code is 60 (functionality) 40 (Quality of Haskell code).
I may interview any student to check that the code is their own (if not fully referenced) and that you fully understand the code and how it works.