COMP105 Lecture 18

Voting Examples

Voting: first past the post

In a first past the post election, whoever gets the **most votes** wins

```
ghci> winner ["red", "blue", "red", "red", "green"]
"red"
```

Getting the candidates

First we need to figure out who the candidates are

```
uniq [] = []
uniq (x:xs) = x : uniq (filter (/=x) xs)
```

```
ghci> uniq ["red", "red", "blue", "green", "red", "blue"]
["red","blue","green"]
```

Counting the votes

This function counts the number of votes for a particular candidate

```
count x list = length (filter (==x) list)
```

```
ghci> count "red" ["red", "blue", "red", "red", "blue"]
3
```

Vote totals

```
totals votes =
    let
        candidates = uniq votes
        f = (\ c -> (count c votes, c))
    in
        map f candidates
```

```
ghci> totals ["red", "blue", "red", "red", "blue"]
[(3,"red"),(2,"blue")]
```

Finding the winner

Recall: tuples are ordered **lexicographically**

```
ghci> max (3, "red") (2, "blue")
(3,"red")
```

```
ghci> maximum [(3, "red"), (2, "blue"), (4, "green")]
(4,"green")
```

Finding the winner

```haskell
winner votes = snd . maximum . totals $ votes
```

```
ghci> winner ["red", "blue", "red", "red", "green"]
"red"
```

Alternative vote

In the **alternative vote** system, voters rank the candidates

- In each round, the candidate with the least number of first preference votes is eliminated
- The winner is the last candidate left once all others have been eliminated

```
ghci> let votes = [["red", "blue", "green"],
                   ["blue", "green"],
                   ["green", "red"],
                   ["blue", "red"],
                   ["red"]]

ghci> av_winner votes
"red"
```

Getting the first choice votes

```
first_choice votes = map head votes
```

```
ghci> let votes = [["red", "blue", "green"],
                   ["blue", "green"],
                   ["green", "red"],
                   ["blue", "red"],
                   ["red"]]
```

```
ghci> first_choice votes
["red","blue","green","blue","red"]
```

Ranking the candidates

```
import Data.List

rank votes = (sort . totals . first_choice) votes
```

```
ghci> let votes = [["red", "blue", "green"],
                   ["blue", "green"],
                   ["green", "red"],
                   ["blue", "red"],
                   ["red"]]
```

```
ghci> rank votes
[(1,"green"),(2,"blue"),(2,"red")]
```

Removing a losing candidate

```haskell
remove_cand c votes =
let
    rm_votes = map (filter (/=c)) votes
    rm_empty = filter (/=[]) rm_votes
in
    rm_empty
```

```
ghci> remove_cand "green" votes
[["red","blue"],["blue"],["red"],["blue","red"],["red"]]
```

```
ghci> remove_cand "red" votes
[["blue","green"],["blue","green"],["green"],["blue"]]
```

Putting it all together

```
av_winner votes =
    let
        ranked = rank_candidates votes
        first = head ranked
    in
        if length ranked == 1
        then first
        else av_winner (remove_cand first votes)
```

```
ghci> av_winner votes
"red"
```