Gequoteerde zitting Haskell: Rijmende zinnen: The east beast feasts least

NAAM: RICHTING:

Enkele praktische afspraken

- Je krijgt twee uur om deze opdracht **individueel** op te lossen.
- Je raadpleegt enkel de Haskell slides (eventueel in afgedrukte vorm met handgeschreven nota's), de cursustekst en de oefeningen die via Toledo voor dit vak ter beschikking gesteld zijn. Je mag ook de manuals vermeld op Toledo en eventueel Hoogle raadplegen.
- In de map **1617_Gequoteerde/Haskell_Donderdag** op Toledo vind je de bestanden `input.txt`, `RhymesSkeleton.hs` en `RhymesSkeleton.hs`. Ook de indienmodule staat daar.
  - **Download en open** het bestand `RhymesSkeleton.hs`, hierin staat reeds een template voor je oplossing.
  - Als eerste vul je bovenaan je naam, studentennummer en richting in.

    ```
    -- Jan Jansen
    -- r0123456
    -- master cw
    ```

  - Je mag **extra functies en types importeren**!
  - Voor elke opdracht zijn een aantal functies reeds gedefinieerd met een bijbehorende typesignatuur. Deze typesignatuur mag niet gewijzigd worden. Vervang telkens `undefined` met jouw implementatie. Je mag argumenten voor het gelijkheidsteken zetten of weghalen. Het is natuurlijk ook altijd toegestaan om extra (hulp)functies te schrijven.
  - Je kan je oplossing testen m.b.v. `RhymesTest.hs`. Dit doe je door in de map waarin de twee `.hs` bestanden staan het volgende commando uit te voeren:

    ```
    runhaskell RhymesTest.hs
    ```

    N.B. dat alle testen slagen betekent niet per se dat je programma helemaal correct is of dat je het maximum van de punten verdient.
  - Na twee uur of wanneer je klaar bent, dien je het bestand `RhymesSkeleton.hs` in via Toledo.

Rijmende zinnen: The east beast feasts least

12 december 2016

In deze opgave zullen we een geannoteerde woordenlijst gebruiken om rijmende zinnen te construeren. Hieronder zie je een mogelijke interactie met het systeem dat we zullen ontwikkelen:

```
What word do you want to rhyme with?
> rhyme
rhyme has 1 syllable(s)
I found the following rhyme words:
CHIME CLIMB CRIME DIME GRIME LIME MIME PRIME RHYME SUBLIME TIME
Here are some sentences you can make using these words:
the lime chime climbs prime
the lime chime climbs sublime
the lime chime climbs time
the lime chime grimes prime
the lime chime grimes sublime
the lime chime grimes time
the lime chime mimes prime
the lime chime mimes sublime
the lime chime mimes time
the lime chime primes sublime
```

1 Inlezen van de data

Elk woord zal voorgesteld worden als een object van het type `Entry`, dit object moet op zijn minst de volgende attributen bevatten:

- Het woord zelf.
- De fonemen, dit is een representatie van de uitspraak van het woord. Elke foneem bestaat uit een aantal letters en eventueel een cijfer, in opgave 2 zullen we deze verder gebruiken. We stellen elke foneem voor door een `String`, de fonemen van een woord wordt dus voorgesteld door een lijst van `String`s.
- Het soort woord. We beschouwen 3 soorten woorden: bijvoeglijk naamwoord (adjective), zelfstandig naamwoord (noun), werkwoord (verb). Stel dit voor met een extra enumeratie-type `WordKind`.

- De vervoegingen/verbuigingen van het woord. Deze mag je ook voorstellen als een lijst van `String`s.

In het invoerbestand beschrijft elke lijn 1 woord. Elke lijn ziet eruit als volgt. Het eerste woord is het woord dat beschreven wordt. Daarna staan de fonemen, gescheiden van elkaar door een liggend streepje (deze kunnen geen spatie bevatten). Daarna staat een letter die het soort woord beschrijft. A voor bijvoeglijk naamwoord, N voor zelfstandig naamwoord en V voor werkwoord. Als laatste komen een aantal vervoegingen/verbuigingen van het woord, gescheiden door een spatie. Het zelfstandig naamwoord *program* wordt dus als volgt beschreven:

```
PROGRAM P-R-OW1-G-R-AE2-M N programs
```

Opgave 1: Maak het datatype `Entry`, voorzie het bestaan van een `Eq`-instantie (een automatisch gegenereerde `Eq`-instantie is goed) en schrijf een `Show`-instantie voor `Entry` en `WordKind` zodat het printen van een `Entry` er exact zo uitziet als een lijn volgens het invoerformaat
Opgave 2: schrijf een functie `parseEntry :: String -> Entry`, die een `String` volgens het invoerformaat omzet naar een `Entry`. Het is waarschijnlijk nuttig om hiervoor een hulpfunctie te schrijven dat een `String` splitst op basis van een bepaald karakter.

2 Lettergrepen en rijmen

De klinker van elke lettergreep bevat als laatste karakter een getal. 0 betekent: er is geen nadruk op deze lettergreep, 1 betekent: dit is de primaire klemtoon, 2 betekent: dit is een secundaire klemtoon van dit woord. Het aantal lettergrepen is dus gelijk aan het aantal cijfers dat voorkomt in de fonemen.

We zeggen dat twee woorden rijmen als alle fonemen vanaf de primaire klemtoon overeenkomen. De woorden *heave* en *achieve* rijmen dus, omdat ze beide beide eindigen op de fonemen "IY1-V". Ze rijmen niet met het woord *achiever*, want *achiever* eindigt met een extra foneem.

```
HEAVE HH-IY1-V N heaves
ACHIEVE AH0-CH-IY1-V V achieved achieving achieves
ACHIEVER AH0-CH-IY1-V-ER0 N achievers
```

Opgave 3: Schrijf een functie `countSyllables :: Entry-> Int` , dat het aantal lettergrepen in een entry telt
Opgave 4: Schrijf een functie `rhymes :: Entry -> Entry -> Bool`, die uitzoekt of twee entry's rijmen met elkaar

We willen graag zinnen genereren met zo verschillend mogelijke woorden. Daarom zullen we afdwingen dat geen enkel woord in onze zin een prefix is van een ander woord.

Opgave 5: Schrijf de functie `noPrefix :: Eq a => [[a]] -> Bool`, die nakijkt of er in de lijst twee elementen voorkomen die prefixen zijn van elkaar. Een lijst is een prefix van een andere lijst, als de andere lijst kan geschreven worden als de eerste lijst, met daarna een extra stuk erna.

3 Zinnen genereren

De zinnen die we willen genereren zijn van de volgende vorm:

"the " , een bijvoeglijk naamwoord (A), een zelfstandig naamwoord (N), een werkwoord (V), een tweede bijvoeglijk naamwoord (A2).

Om het werkwoord grammaticaal te laten kloppen moet we niet het werkwoord zelf in de zin hebben maar de laatste vervoeging nemen uit de lijst van vervoegingen. Verder moet gelden dat het eerste bijvoeglijk naamwoord lexicografisch kleiner is dan het tweede (gebruik hiervoor de standaard orde op `String`s). En er moet gelden dat geen enkel van deze woorden (behalve het lidwoord "the") een prefix is van een ander woord.

Zorg ook dat de volledige zin in lowercase letters staat, gebruik hiervoor de functie `toLower` uit `Data.Char`. De analoge functie `toUpper` uit `Data.Char`, dient om letter om te zetten naar zijn uppercase versie, zal verderop ook nog nuttig zijn.

Opgave 6: Schrijf de functie `makeSentence :: [Entry] -> [String]`, die uit een aantal entries, alle mogelijke zinnen van bovenstaande vorm maakt.

Nu zijn we klaar om onze hoofdfunctie te schrijven. Deze start met het inlezen van een bestand "input.txt"[1] en deze te parsen naar entries. Hij stelt vervolgens de vraag "What word do you want to rhyme with?". Vervolgens kan de gebruiker een woord intypen. Het programma zet dit woord om naar hoofdletters en zoekt naar dit woord in de lijst van entries. Als het woord niet in de lijst van entries voorkomt dan antwoordt hij met "I couldn't find the word" en stopt het programma. Als het woord wel voorkomt in de lijst van entries, dan worden er de volgende dingen geprint:

1. *het woord* "has" *het aantal lettergrepen* "syllable(s)"
2. "I found the following rhyme words:"
3. *Een lijst van alle UNIEKE rijmwoorden uit "input.txt", gescheiden door spaties. Let erop dat sommige woorden meerdere keren kunnen voorkomen in "input.txt" (bijvoorbeeld woorden die zowel als zelfstandig naamwoord als als werkwoord kunnen voorkomen)*
4. "Here are some sentences you can make using these words:"
5. *De 10 eerste zinnen gegenereerd door `makeSentence` toegepast op de lijst van rijmwoorden*

Opgave 7: Schrijf de functie `run :: IO ()` zoals hierboven beschreven

Het kan nuttig zijn om onderstaande functie te gebruiken:

- `words :: String -> [String]`, zet een string met meerdere woorden die gescheiden zijn door een spatie om, naar een lijst van de individuele woorden.
- `unwords :: [String] -> String`, de inverse van words, zet spaties tussen elk woord van de lijst en maakt er 1 String van.

[1] Gebruik hiervoor de functie `readFile :: String -> IO String`. Deze functie neemt de naam van een bestand en geeft de inhoud ervan terug.

- `lines :: String -> [String]`, zet een String met meerdere lijnen om naar een lijst van Strings, elk element van de lijst komt dan overeen met een lijn uit de originele String.
- `unlines :: [String] -> String`, de inverse van lines, zet newlines tussen elke string in de lijst, maakt 1 een grote String van.

```
What word do you want to rhyme with?
> fail
fail has 1 syllable(s)
I found the following rhyme words:
AIL ALE ASSAIL AVAIL BAIL BALE CARRELL CURTAIL DALE DERAIL DETAIL
   EMAIL ENTAIL EXHALE FAIL FLAIL FRAIL GALE HAIL HALE IMPALE
   INHALE JAIL MAIL MALE NAIL PAIL PALE PREVAIL QUAIL RAIL SAIL
   SALE SCALE SNAIL STALE TAIL TALE TRAIL TRAVAIL UNVEIL VALE VEIL
    WAIL WALE WHALE
Here are some sentences you can make using these words:
the frail ale ails hale
the frail ale ails mail
the frail ale ails male
the frail ale ails pale
the frail ale ails stale
the frail ale ails tail
the frail ale assails hale
the frail ale assails mail
the frail ale assails male
the frail ale assails pale
```