Haskell: Word Wrap

NAAM: RICHTING:

Enkele praktische afspraken

- Op Toledo en esystant vind je de bestanden `MyHaskell.hs`, `MyHaskellTest.hs` en `Testing.hs`
  - In het bestand `MyHaskell.hs` 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 ook **extra functies en types importeren**.
  - Voor elke opdracht zijn een aantal functies en typeklassen reeds gedefinieerd met bijbehorende typesignaturen. De functies en typesignaturen **mag je niet aanpassen**. Vervang telkens `error "..."` met jouw implementatie. Je mag argumenten voor het gelijkheidsteken zetten, maar probeer indien mogelijk de functie *point-free* te schrijven. Het is natuurlijk ook altijd toegestaan om extra (hulp)functies te schrijven.
  - Je mag gebruik maken van de slides op Toledo, van e-systant, hoogle (http://www.haskell.org/hoogle/) en de bijbehorende hackage documentatie.
  - Ook al vormt deze opgave een geheel, je kan de meeste opdrachten afzonderlijk maken. Dus **wanneer je vastzit op een opdracht, probeer de volgende**.
  - Je kan je oplossing testen m.b.v. `MyHaskellTest.hs`. Dit doe je door in de map waarin de drie `.hs`-bestanden staan het volgende commando uit te voeren:

    ```
    runhaskell MyHaskellTest.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.

- De oplossing (`MyHaskell.hs`) dien je in op Toledo.

Word Wrap

In deze opgave gaan we een tekst opdelen in lijnen met een maximale lengte zodat de tekst op een pagina met een bepaalde breedte past.

Neem bijvoorbeeld de onderstaande tekst.

> Leverage agile frameworks to provide a robust synopsis for high level overviews. Iterative approaches to corporate strategy foster collaborative thinking to further the overall value proposition. Organically grow the holistic world view of disruptive innovation via workplace diversity and empowerment.

Deze doorlopende tekst is gesplitst in lijnen door de tekstverwerker waarmee deze opgave is opgesteld, namelijk LaTeX. LaTeX heeft een slim algoritme om lijnen te splitsen, waarbij de breedte van de tekst zo gelijk mogelijk wordt gehouden en woorden intelligent gesplitst worden. In deze opgave houden we het simpel: we splitsen geen woorden en we volgen een eenvoudig *greedy* algoritme waarbij zoveel mogelijk woorden op een lijn worden geplaatst totdat er geen plaats meer is op de lijn, waarna we overgaan naar de volgende lijn totdat er geen woorden meer overblijven. Wanneer de originele tekst een newline bevat, zal deze newline op deze plaats in de gesplitste tekst voorkomen (denk aan \\ in LaTeX).

Als we dit algoritme uitvoeren met de bovenstaande tekst en als lijnbreedte 50 krijgen we de onderstaande schikking. **N.B.** de bovenste lijn en de rechter kantlijn zijn toegevoegd aan deze figuur om het tellen te vergemakkelijken. Je oplossing zal deze dus **niet** moeten tekenen.

```
----+----1----+----2----+----3----+----4----+----5|
Leverage agile frameworks to provide a robust     |
synopsis for high level overviews. Iterative      |
approaches to corporate strategy foster           |
collaborative thinking to further the overall     |
value proposition. Organically grow the holistic  |
world view of disruptive innovation via workplace |
diversity and empowerment.                        |
```

Wanneer we het algoritme uitvoeren met lijnbreedte 32 krijgen we de onderstaande schikking:

```
----+----1----+----2----+----3--|
Leverage agile frameworks to    |
provide a robust synopsis for   |
high level overviews. Iterative |
approaches to corporate strategy|
foster collaborative thinking to|
further the overall value       |
proposition. Organically grow   |
the holistic world view of      |
disruptive innovation via       |
workplace diversity and         |
empowerment.                    |
```

Aanpak

De aanpak in deze opdracht is als volgt: we splitsen de originele tekst (`String`) in een lijst van `LineItem`s, een nieuw datatype. Het opdelen in lijnen doen we d.m.v. een aantal transformaties van lijsten van `LineItem`s. Ten slotte zetten we deze lijsten van `LineItem`s terug om in een `String`.

Deel 1: Line Items

Opdracht 1.1: Definieer een datatype genaamd `LineItem`. Een `LineItem` is ofwel een spatie (`" "`), een newline (`"\n"`), ofwel een woord (een `String`). Een woord mag geen spaties of newlines bevatten, maar wel leestekens. Meerdere spaties in de invoer zullen resulteren in meerdere spatie `LineItem`s, idem voor newlines. **Denk goed na over wat de constructoren van een spatie en een newline moeten bijhouden.**

Zie opdracht 1.3 voor enkele voorbeelden van `LineItem`s.

Opdracht 1.2: Implementeer de functies `mkSpace :: LineItem`, `mkNewline :: LineItem`, en `mkWord :: String -> LineItem` die de respectievelijke `LineItem`s maken.

Opdracht 1.3: Maak een instantie van `Show` voor `LineItem`.

Een spatie wordt getoond als `" "`, een newline als `"\n"`, en een *woord* als `"woord"`. **Let op:** de aanhalingstekens zijn deel van de `String`.

Hint: `show "foo" = "\"foo\""`.

Voorbeelden:

```
> mkSpace
" "
> mkNewline
"\n"
> mkWord "Hello"
"Hello"
> map show [mkSpace, mkNewline, mkWord "Hello"]
["\" \"","\"\\n\"","\"Hello\""]
```

Opdracht 1.4: Implementeer de functie `toLineItems :: String -> [LineItem]` die een `String` opsplitst in een lijst van `LineItem`s.

Voorbeelden:

- De zin "`See ya, John.`" bestaat uit 5 `LineItem`s: `["See"," ","ya","," ","John."]`.
- De zin "`Hint: read the\nassignment carefully!`" bestaat uit 9 `LineItem`s: `["Hint:"," ","read"," ","the","\n","assignment"," ","carefully!"]`.
- De zin "`! oops  \n\n `" bestaat uit 8 `LineItem`s: `["!"," ","oops"," "," ","\n","\n"," "]`.

Opdracht 1.5: Implementeer de functie `fromLineItems :: [LineItem] -> String` die een lijst van `LineItem`s samenvoegt tot een `String`.

Eigenschap: voor elke `String s`, geldt dat `fromLineItems (toLineItems s) == s`.

Deel 2: Word Wrap

Beschrijving van het algoritme:

- Plaats zoveel mogelijk woorden op een lijn totdat er geen plaats meer is op de lijn, begin daarna een nieuwe lijn. Herhaal dit tot er geen woorden meer overblijven.
- Als er een newline staat in de invoer, moet deze op dezelfde plaats voorkomen in de uitvoer.
- Meerdere opeenvolgende spaties worden vervangen door één spatie.
- Spaties op het begin of einde van een lijn worden weggelaten.
- Geen enkele lijn mag langer zijn dan de maximale lijnbreedte tenzij een woord langer is dan de maximale lijnbreedte. In dat geval zal dat woord alleen staan op een lijn.

We splitsen het algoritme op in verschillende eenvoudigere stappen.

1. Eerst verwijderen we alle spaties. Omdat we weten dat tussen elke twee woorden één spatie moet komen, hoeven we deze niet meer expliciet bij te houden, wat de rest van de implementatie makkelijker maakt. Dit zorgt er ook voor dat meerdere opeenvolgende spaties vervangen worden door één spatie, zoals gevraagd in de regels. Daarnaast zorgt dit er ook voor dat er geen spaties op het begin of einde van een lijn overblijven.
2. Vervolgens splitsen we de tekst op in lijnen, let op: dit zijn de lijnen aangegeven door de newlines die reeds aanwezig zijn in de tekst. We kunnen dan elk van die lijnen afzonderlijk splitsen wanneer deze te lang worden, zonder te moeten nadenken over newlines in de originele tekst.
3. Daarna gaan we woorden die langer zijn dan de maximale lijnbreedte al op hun eigen lijn zetten. Bij het eigenlijke splitsen hoeven we er dan niet meer voor te zorgen dat er geen andere woorden voor dit woord komen op dezelfde lijn.
4. Dan gaan we het eigenlijke opdelen van de lijnen doen: we splitsen een reeks van woorden op in lijnen door zoveel mogelijk woorden op een lijn te zetten totdat er geen plaats meer is op de lijn. Hierbij houden we rekening met de breedte van de spatie tussen elke twee woorden.
5. Vervolgens zetten we terug spaties tussen de woorden in elke lijn.
6. Daarna voegen we de lijnen samen door newlines tussen elke lijn te zetten.

Opdracht 2.1: Implementeer de functie `removeSpaces :: [LineItem] -> [LineItem]` die alle spaties uit een lijst van `LineItem`s verwijdert.

Voorbeelden:

```
> removeSpaces [mkWord "hello", mkSpace, mkWord "world", mkNewline, mkWord "Bye", mkSpace]
["hello","world","\n","Bye"]
> removeSpaces [mkWord "hi", mkSpace, mkSpace, mkNewline, mkSpace, mkNewline, mkSpace]
["hi","\n","\n"]
```

Opdracht 2.2: Implementeer de functie `splitInLines :: [LineItem] -> [[LineItem]]` die een lijst van `LineItem`s splitst op alle plaatsen waar een newline voorkomt. In het resultaat zullen geen newlines meer zitten.

Je mag aannemen dat in de invoer geen spaties zitten.

Voorbeelden:

```
> splitInLines [mkWord "hi", mkNewline, mkWord "bye"]
[[mkWord "hi"], [mkWord "bye"]]
> splitInLines [mkNewline, mkWord "hi", mkNewline, mkNewline, mkWord "bye", mkNewline]
[[], [mkWord "hi"], [], [mkWord "bye"], []]
> splitInLines []
[[]]
> splitInLines [mkNewline]
[[], []]
> splitInLines [mkNewline, mkNewline]
[[], [], []]
> splitInLines [mkWord "foo", mkNewline]
[[mkWord "foo"], []]
```

Opdracht 2.3: Implementeer de functie `separateTooLongWords :: Int -> [LineItem] -> [[LineItem]]` die een lijst van `LineItem`s splitst bij elk woord dat langer is dan de maximale lijnbreedte. Elke lijst van `LineItem`s stelt een lijn voor, dus deze functie splitst één lijn op in een lijst van lijnen. Bij een woord dat te lang is, wordt de lijn opgedeeld in drie lijnen (lijsten): een lijn van de woorden voor het te lange woord, een lijn met enkel het te lange woord en een lijn met de woorden na het te lange woord. Wanneer er geen woorden voor het te lange woord komen, wordt de eerste lijst weggelaten. Wanneer er geen woorden na het te lange woord komen, wordt de derde lijst weggelaten. Dit wordt herhaald voor elk woord dat te lang is.

Je mag aannemen dat de invoer enkel woorden bevat, geen spaties of newlines.

Voorbeelden:

```
> separateTooLongWords 6 [mkWord "look", mkWord "a", mkWord "brontosaurus",
                          mkWord "over", mkWord "there"]
[["look","a"],["brontosaurus"],["over","there"]]
> separateTooLongWords 3 [mkWord "Yuuuuge", mkWord "amazing"]
[["Yuuuuge"],["amazing"]]
```

```
> separateTooLongWords 3 [mkWord "Banana"]
[["Banana"]]
> separateTooLongWords 100 [mkWord "Banana"]
[["Banana"]]
```

Opdracht 2.4: Implementeer de functie `wrap :: Int -> [LineItem] -> [[LineItem]]` die een lijst van `LineItem`s telkens splitst wanneer de maximale lijnbreedte bereikt is. Elke lijst van `LineItem`s stelt een lijn voor, dus deze functie splitst één lijn op in een lijst van lijnen. Herinner je het algoritme: *plaats zoveel mogelijk woorden op een lijn totdat er geen plaats meer is op de lijn, begin daarna een nieuwe lijn.* Vergeet niet rekening te houden met de spatie tussen twee woorden. Vergeet ook niet dat sommige woorden langer dan de maximale lijnbreedte kunnen zijn.

Je mag aannemen dat de invoer enkel woorden bevat, geen spaties of newlines.

Voorbeelden:

```
> wrap 7 [mkWord "foo", mkWord "bar", mkWord "qu", mkWord "u", mkWord "x", mkWord "banana"]
[["foo","bar"],["qu","u","x"],["banana"]]
> wrap 4 [mkWord "gyronef"]
[["gyronef"]]
```

Opdracht 2.5: Implementeer de functie `joinLineWithSpaces :: [LineItem] -> [LineItem]` die een spatie tussen elke twee woorden in de lijst van `LineItem`s zet.

Je mag aannemen dat de invoer enkel woorden bevat, geen spaties of newlines.

Voorbeelden:

```
> joinLineWithSpaces [mkWord "so", mkWord "much", mkWord "space"]
["so"," ","much"," ","space"]
```

Opdracht 2.6: Implementeer de functie `joinLinesWithNewlines :: [[LineItem]] -> [LineItem]` die de gegeven lijst van lijnen samenvoegt tot één lijst van `LineItem`s waarbij elke lijn gescheiden wordt door een newline.

Je mag aannemen dat de invoer enkel woorden en spaties bevat, geen newlines.

Voorbeelden:

```
> joinLinesWithNewlines [[mkWord "hi", mkSpace, mkWord "there"],[mkWord "bye"]]
["hi"," ","there","\n","bye"]
```

Resultaat In de functie `wordWrap` worden de bovenstaande functies op de juiste wijze gecombineerd. Deze functie is gegeven en hoef je dus zelf niet te schrijven. Met deze functie kan je makkelijk testen of je het algoritme correct hebt geïmplementeerd a.d.h.v. de onderstaande voorbeelden.

Voorbeelden:

- ```
  > wordWrap 5 "a b c d e f g"
  "a b c\nd e f\ng"
  > wordWrap 4 "a b c d e f g"
  ```
  ```

```
"a b\nc d\ne f\ng"
> wordWrap 5 "a\nb c d e f g"
"a\nb c d\ne f g"
> wordWrap 4 "\n food"
"\nfood"
> wordWrap 4 "a b c delta e f g"
"a b\nc\ndelta\ne f\ng"
> wordWrap 7 "foo  bar  "
"foo bar"
> wordWrap 20 "  foo  \n  \n  bar  "
"foo\n\nbar"
```

- Zie de twee voorbeelden uit de inleiding (`putStrLn $ wordWrap 50 "..."` en `putStrLn $ wordWrap 32 "..."`).

Hint: met de testsuite kan je al de bovenstaande voorbeelden in één keer uitvoeren.

Deel 3: Interactief Word Wrappen

Opdracht 3.1: Implementeer de functie `getLines :: IO String` die een tekst (`String`) die newlines kan bevatten, inleest. De gebruiker geeft de tekst lijn per lijn in, waarbij elke lijn wordt beëindigd met een Enter. Dit gaat door totdat de gebruiker `STOP` ingeeft (gevolgd door Enter).
Hint: gebruik `getLine` om één lijn in te lezen. Vergeet de newline niet na elke lijn.

Voorbeelden: De lijn na `STOP` is telkens het resultaat van de `getLines`-oproep.

```
> getLines
Hello
STOP
"Hello\n"
> getLines
Hello, there

Bye now
STOP
"Hello, there \n\nBye now\n"
```

Opdracht 3.2: Implementeer de functie `interactiveWrapper :: IO ()` die de gebruiker een lijnbreedte vraagt en een tekst, waarna de tekst wordt 'gewrapt' volgens de ingelezen lijnbreedte. Lees de tekst in met de `getLines` functie uit de vorige opdracht. Je mag aannemen dat de gebruiker een geldig getal ingeeft voor de lijnbreedte.

Voorbeelden:

```
> main
Please enter a line width: 4
```

```
Please enter a text to wrap:
a
b c d e f g
STOP
a
b c
d e
f g
```