Practicumopdracht 1: Elektronische agenda's

Het practicum Functioneel programmeren staat dit jaar in het teken van elektronische agenda's. Deze eerste van in totaal drie opdrachten bestaat uit het implementeren van een aantal eenvoudige operaties op agenda's. De opdracht wordt uitgevoerd in groepjes van maximaal twee studenten.

Startbestand

Op de wikipagina van het practicum tref je een startbestand aan (`http://www.cs.uu.nl/wiki/FP/Practicum`). Dit Haskell-bestand, `Agenda.hs` genaamd, bevat de startcode voor de opdracht: typesynoniemen voor elektronische agenda's en taken, een functie om agenda's mee naar het scherm te schrijven en een aantal voorbeeldagenda's. Het is de bedoeling dat je dit bestand uitbreidt met je eigen functiedefinities.

Inleveren

Vermeld aan het begin van het bestand – in commentaarregels – je naam, login en studentnummer. (Als je de opdracht met zijn tweeën doet, vermeld dan de gegevens van beide studenten.) De opdracht bestaat uit acht opgaven. Zorg dat de namen en types van je functies overeenkomen met wat in de opgaven gevraagd wordt. Dit voorkomt misverstanden bij het nakijken. Als je klaar bent met de opdracht, lever je je uitbreiding van het bestand `Agenda.hs` in via Submit (`http://www.cs.uu.nl/docs/vakken/submit`). De deadline is op **maandag 6 maart 2006** om **23:59 uur**. Zorg ervoor dat de ingeleverde code werkt met de laatste versie van Helium (versie 1.6).

Tips

- Deel je code op in kleine functies. Je programma wordt daardoor overzichtelijk. Bovendien kunnen kleine hulpfuncties vaak op meerdere plekken gebruikt worden.
- Maak gebruik van de speciale syntaxis voor lijsten. Met behulp van lijstcomprehensies kunnen functies op lijsten vaak verrassend eenvoudig opgeschreven worden.
- Maak gebruik van standaardfuncties. Deze kunnen je veel werk uithanden nemen. In de startcode wordt de bibliotheek *List* geïmporteerd. Zoek uit welke functies deze bibliotheek bevat en hoe je ze in je programma kunt gebruiken.
- Test je code uitvoerig. Gebruik daarbij niet alleen de voorbeeldagenda's uit het startbestand, maar definieer ook zelf agenda's met interessante gevallen, zodat je je ervan kunt overtuigen dat je functies ook voor invoer met randwaarden correcte resultaten opleveren.
- Maak je programma leesbaar: kies toepasselijke namen voor hulpfuncties, zorg voor een prettige layout en gebruik commentaar om ingewikkelde functiedefinities toe te lichten.

Figuur 1 Collegerooster

```
alleGroepen :: [Deelnemer]
alleGroepen = ["groep 1","groep 2","groep 3","groep 4"]

collegerooster :: Agenda
collegerooster =
  [(((15,2,2006,13,00),(15,2,2006,15,00)),"werkcollege",["theorie" ],["groep 1"])
  ,(((15,2,2006,13,00),(15,2,2006,15,00)),"werkcollege",["theorie" ],["groep 2"])
  ,(((15,2,2006,13,00),(15,2,2006,15,00)),"werkcollege",["theorie" ],["groep 3"])
  ,(((15,2,2006,13,00),(15,2,2006,15,00)),"werkcollege",["theorie" ],["groep 4"])
  ,(((15,2,2006,15,00),(15,2,2006,17,00)),"hoorcollege",["theorie" ],alleGroepen )
  ,(((17,2,2006,9 ,00),(17,2,2006,11,00)),"practicum"  ,["praktijk"],alleGroepen)
  ,(((17,2,2006,11,00),(17,2,2006,13,00)),"werkcollege",["theorie" ],["groep 1"])
  ,(((17,2,2006,11,00),(17,2,2006,13,00)),"werkcollege",["theorie" ],["groep 2"])
  ,(((17,2,2006,11,00),(17,2,2006,13,00)),"werkcollege",["theorie" ],["groep 3"])
  ,(((17,2,2006,11,00),(17,2,2006,13,00)),"werkcollege",["theorie" ],["groep 4"])
  ,(((17,2,2006,13,00),(17,2,2006,15,00)),"hoorcollege",["theorie" ],alleGroepen )
  ]
```

Agenda's representeren

Elektronische agenda's worden gerepresenteerd met behulp van de volgende types:

```
type Tijdstip      = (Int, Int, Int, Int, Int)
type Tijdslot      = (Tijdstip, Tijdstip)
type Omschrijving = String
type Categorie     = String
type Deelnemer     = String
type Afspraak      = (Tijdslot, Omschrijving, [Categorie], [Deelnemer])
type Agenda        = [Afspraak].
```

Een tijdstip is een 5-tupel waarvan de componenten achtereenvolgens dagen, maanden, jaren, uren en minuten voorstellen. Een voorbeeld van een waarde van type Tijdstip is $(15, 2, 2006, 13, 00)$, ofwel 15 februari 2006, 13:00 uur. Een begintijd en een eindtijd – dat wil zeggen, twee expressies van type Tijdstip – vormen samen een tijdslot. Behalve een tijdslot, houden we voor elke afspraak een omschrijving, een lijst van categorieën en een lijst van deelnemers bij; voor het weergeven van omschrijvingen, categorieën en deelnemers gebruiken we strings. Een agenda is simpelweg een lijst van afspraken.

Voorlopig gaan we ervan uit dat agenda's uit geldige afspraken bestaan: dus dat waarden van het type Tijdstip ook echt overeen komen met geldige datum-tijdcombinaties en dat er geen afspraken zijn die eerder eindigen dan beginnen.

Figuur 1 toont een voorbeeld van een agenda: het collegerooster voor het vak Functioneel programmeren in week 7.

Agenda's weergeven

Het startbestand bevat een functie *toonAgenda* :: Agenda $\rightarrow$ IO $()$. Deze functie, die een agenda overzichtelijk weergegeven naar het scherm schrijft, kun je gebruiken bij het testen van je functies.

Deelagenda's

Een van de voordelen van een elektronische agenda ten opzichte van een papieren agenda is het gemak waarmee relevante informatie geselecteerd kan worden. Voor een student die het het vak Functioneel programmeren volgt, is het complete collegerooster uit figuur 1 bijvoorbeeld niet zo interessant: het is vooral belangrijk te weten wanneer de colleges voor de eigen werkcollegegroep zijn. Een veelvoorkomende operatie op agenda's is dan ook het selecteren van afspraken op basis van een bepaald criterium. De afspraken die aan zo'n criterium voldoen, vormen dan samen een zogenaamde deelagenda.

Opgave 1. Schrijf een functie *voorDeelnemer* :: Agenda → Deelnemer → Agenda, zodat *agenda* `` `voorDeelnemer` `` *deelnmr* een deelagenda oplevert waarin precies die afspraken uit *agenda* zijn opgenomen waaraan deelnemer *deelnmr* deelneemt. Bijvoorbeeld: de expressie *collegerooster* `` `voorDeelnemer` `` `"groep 1"` levert een agenda op met vijf afspraken (twee hoorcolleges, twee werkcolleges en het practicum; zie figuur 1). □

Opgave 2. Schrijf een functie *voorCategorie* :: Agenda → Categorie → Agenda, zodat *agenda* `` `voorCategorie` `` *cat* een deelagenda oplevert waarin precies die afspraken uit *agenda* zijn opgenomen die geplaatst zijn in de categorie *cat*. Bijvoorbeeld: de expressie *collegerooster* `` `voorCategorie` `` `"theorie"` levert een agenda op met tien afspraken (de hoor- en werkcolleges; zie figuur 1). □

Opgave 3. Schrijf een functie *voorInterval* :: Agenda → (Tijdstip, Tijdstip) → Agenda, zodat *agenda* `` `voorInterval` `` $(begin, eind)$ een deelagenda oplevert die precies die afspraken uit *agenda* bevat die op of na het tijdstip *begin* beginnen en voor of op het tijdstip *eind* eindigen. Bijvoorbeeld: de expressie *collegerooster* `` `voorInterval` `` $((15,2,2006,14,00),(17,2,2006,11,00))$ levert een agenda op met twee afspraken (het eerste hoorcollege en het practicum; zie figuur 1). □

Behalve operaties als *voorDeelnemer*, *voorCategorie* en *voorInterval*, die een enkele deelagenda produceren, kunnen we ook functies schrijven die meerdere (en mogelijk overlappende) deelagenda's opleveren.

Opgave 4. Schrijf een functie *perDeelnemer* :: Agenda → [Agenda], zodat *perDeelnemer agenda* voor elke deelnemer in *agenda* een deelagenda oplevert. Bijvoorbeeld: de expressie *perDeelnemer collegerooster* levert een lijst met vier deelagenda's op – één voor elke werkcollegegroep (zie figuur 1) – die elk vijf afspraken bevatten (de twee hoorcolleges, het practicum en de twee werkcolleges voor de betreffende groep). □

Opgave 5. Schrijf een functie *perJaar* :: Agenda → [Agenda], zodat *perJaar agenda* voor elk jaar in *agenda* een deelagenda oplevert, waarbij afspraken die een of meer jaargrenzen overschrijden gesplitst zijn. Bijvoorbeeld: voor de agenda *jaarrooster* in figuur 2 levert de expressie *perJaar jaarrooster* de volgende lijst met deelagenda's op:

```
[
  [(((8 ,9 ,2005,0,0),(11,11,2005,23,59)),"periode 1",[],[])
  ,(((14,11,2005,0,0),(31,12,2005,23,59)),"periode 2",[],[])
  ]
```

Figuur 2 Jaarrooster

```
jaarrooster :: Agenda
jaarrooster =
  [(((8 ,9 ,2005,0,0),(11,11,2005,23,59)),"periode 1",[],[])
  ,(((14,11,2005,0,0),(3 ,2 ,2006,23,59)),"periode 2",[],[])
  ,(((6 ,2 ,2006,0,0),(21,4 ,2006,23,59)),"periode 3",[],[])
  ,(((24,4 ,2006,0,0),(7 ,7 ,2006,23,59)),"periode 4",[],[])
  ]
```

Figuur 3 Een vreemde agenda

```
vreemd :: Agenda
vreemd =
  [(((32,3,2006,11,64),(32,3,2006,12,30)),"grap uithalen"           ,[],["miranda"])
  ,(((11,4,2006,21,00),(10,4,2006,13,00)),"tentamen FP voorbereiden",[],["miranda"])
  ]
```

```
,  [(((1 ,1 ,2006,0,0),(3 ,2 ,2006,23,59)),"periode 2",[],[])
   ,(((6 ,2 ,2006,0,0),(21,4 ,2006,23,59)),"periode 3",[],[])
   ,(((24,4 ,2006,0,0),(7 ,7 ,2006,23,59)),"periode 4",[],[])
   ]
].
```

Let met name op de afspraak voor periode 2, die in tweeën gesplitst is: het eerste deel is opgenomen in de deelagenda voor 2005, het tweede in de deelagenda voor 2006. □

Invoercorrectie

Met de types die we gekozen hebben voor de representatie van tijdstippen en tijdsloten kunnen we merkwaardige agenda's construeren. Bekijk bijvoorbeeld de agenda *vreemd* in figuur 3 eens. De eerste afspraak in deze agenda heeft plaats op 32 maart en begint om 64 minuten over elf! Maar ook de tweede afspraak is niet in orde: deze eindigt nog voor hij begint. In een realistische agenda-applicatie zal gecontroleerd moeten worden of alle afspraken aan geldige tijdsloten zijn toegewezen en in het geval van incorrecte invoer zullen passende maatregelen genomen moeten worden.

Ongeldige tijdstippen moeten bijvoorbeeld herschreven worden: 32 maart 2006 wordt dan 1 april 2006, 33 december 2006 wordt 2 januari 2007, 11:64 uur wordt 12:04 uur, 23:70 wordt 0:10 (op de volgende dag), enzovoort. Schrikkeljaren maken dit proces iets ingewikkelder: 29 februari 2006 wordt 1 maart 2006, maar 29 februari 2008 blijft 29 februari 2008. 2008 is immers een schrikkeljaar. (Schrikkeljaren zijn jaren met een jaartal dat deelbaar is door vier. Een uitzondering vormen jaren met jaartallen deelbaar door honderd: dat zijn geen schrikkeljaren – tenzij het jaartal ook deelbaar is door 400: dan is het weer wel een schrikkeljaar. 1900 en 2100 zijn dus bijvoorbeeld geen schrikkeljaren, maar 2000 en 2400 wel.)

Opgave 6. Schrijf een functie *normaliseer* :: Agenda → Agenda, zodat *normaliseer agenda* een nieuwe agenda oplevert waarin precies die afspraken uit *agenda* zijn

Figuur 4 Histogram

```
theorie  :  ***********  (80.0%)
praktijk :  ****         (20.0%)
```

opgenomen die eerder beginnen dan ze eindigen. Bovendien moeten alle tijdstippen in de nieuwe agenda op de hierboven beschreven manier herschreven worden. (Houd dus ook rekening met schrikkeljaren.) Bijvoorbeeld: de expressie *normaliseer vreemd* levert de agenda

$$[(((1,4,2006,12,4),(1,4,2006,12,30)),\texttt{"grap uithalen"},[\,],[\texttt{"miranda"}])]$$

op.

Let erop dat je tijdstippen op de juiste manier met elkaar vergelijkt. Het tijdslot $((15,3,2006,12,90),(15,3,2006,13,15))$ kan bijvoorbeeld herschreven worden als $((15,3,2006,13,30),(15,3,2006,13,15))$ en eindigt dus eerder dan het begint: een afspraak in dit tijdslot mag daarom niet in de nieuwe agenda worden opgenomen. Bedenk verder hoe je met negatieve getallen omgaat. □

Overzichten

Behalve voor het raadplegen van individuele afspraken en deelagenda's kan een elektronische agenda ook prima gebruikt worden voor het genereren van allerlei overzichten. Een voorbeeld van zo'n overzicht vind je in figuur 4. Daar is een histogram weergegeven van het tijdverbruik per categorie in de deelagenda *collegerooster* '*voorDeelnemer*' `"groep 1"` (zie opgave 1 en figuur 1). Het histogram laat zien dat 80% van de tijd besteed wordt aan activiteiten in de categorie theorie, tegenover 20% voor activiteiten in de categorie praktijk. (Een sterretje in het histogram komt overeen met vijf procent.)

Opgave 7. Schrijf een functie *histogram* :: Agenda $\rightarrow$ String, zodat *histogram agenda* een histogram van het tijdverbruik per categorie in *agenda* oplevert. Zorg ervoor dat de weergave van het histogram overeenkomt met die van het histogram in figuur 4. Let erop dat het aantal afspraken in een bepaalde categorie er niet toe doet; het gaat erom hoe lang de afspraken duren. □

Planning

Naast het verstrekken van informatie over vastgelegde afspraken, kan een agenda-applicatie je ook helpen met het inboeken van nieuwe afspraken. Dat inboeken kan bijvoorbeeld gedaan worden aan de hand van een lijstje taken die binnen een bepaald tijdslot uitgevoerd moeten zijn.

Taken kunnen gerepresenteerd worden met behulp van het volgende type:

type Taak = (Omschrijving, [Categorie], [Deelnemer], Int).

De laatste component van een Taak-tupel stelt het aantal minuten voor dat nodig is voor het uitvoeren van de taak.

Figuur 5 Dagplanning

```
maandag :: Agenda
maandag =
  [(((13,2,2006,9 ,00),(13,2,2006,10,45)),"college",[],["miranda"])
  ,(((13,2,2006,12,45),(13,2,2006,13,15)),"lunch"  ,[],["miranda"])
  ,(((13,2,2006,13,15),(13,2,2006,15,00)),"college",[],["miranda"])
  ]

taken    :: [Taak]
taken    =
  [("boodschappen",[],["miranda"],30 )
  ,("rijles"      ,[],["miranda"],100)
  ,("practicum"   ,[],["miranda"],120)
  ]

interval :: Tijdslot
interval = ((13,2,2006,9,00),(13,2,2006,17,00))
```

Opgave 8. Schrijf een functie *plan* :: Agenda → Tijdslot → [Taak] → (Agenda, [Taak]), zodat *plan agenda interval taken* een paar $(agenda', taken')$ oplevert, waarbij $agenda'$ alle afspraken uit *agenda* bevat, plus zo veel mogelijk van de ingeboekte taken (gemeten in minuten) uit de lijst *taken*. Alle taken moeten ingeboekt worden in het tijdslot *interval*. De volgorde van de taken mag hierbij gewijzigd worden. Een ingeboekte taak mag alleen overlappen met een bestaande afspraak als geen van de deelnemers aan de taak deelneemt aan de afspraak. Taken die niet ingeboekt kunnen worden, moeten worden opgeleverd in de lijst $taken'$ (de tweede component van het resultaat).

Bijvoorbeeld: zie figuur 5. Een mogelijk resultaat voor de expressie *plan maandag interval taken* zou kunnen zijn:

```
([(((13,2,2006,9 ,00),(13,2,2006,10,45)),"college"  ,[],["miranda"])
 ,(((13,2,2006,10,45),(13,2,2006,12,25)),"rijles"    ,[],["miranda"])
 ,(((13,2,2006,12,45),(13,2,2006,13,15)),"lunch"     ,[],["miranda"])
 ,(((13,2,2006,13,15),(13,2,2006,15,00)),"college"   ,[],["miranda"])
 ,(((13,2,2006,15,00),(13,2,2006,17,00)),"practicum",[],["miranda"])
 ]

,[("boodschappen",[],["miranda"],30)]
).
```

□