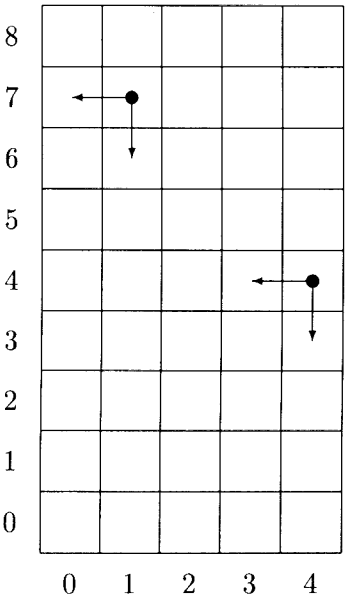
Tentamen
ORIENTATIE INFORMATICA
11 maart 2002
10.00 – 13.00 uur; WSN 20

Opmerkingen vooraf:

- geef bij elke Haskell-functie ook de typering.
- in elk onderdeel mag je gebruik maken van vorige onderdelen, ook als je niet in staat bent geweest deze te maken.
- Niet elke opgave telt even zwaar. Bij elk onderdeel staat aangegeven hoeveel punten er maximaal gescoord kunnen worden. Bij 100 of meer punten is het tentamenresultaat een 10. In andere gevallen is het tentamenresultaat het aantal punten gedeeld door 10.

Opgave 1

In deze opgave bekijken we het volgende bordspel. Het bord bestaat uit een rechthoekig schema van vierkanten, vijf eenheden breed en een willekeurig aantal (> 0) lang. Het spel begint met het plaatsen van een pion op het laatste vierkant van de hoogst genummerde rij.
Een zet bestaat uit het verplaatsen van de pion op de manier zoals in de tekening hiernaast staat aangegeven: of naar het vakje direct onder de huidige positie of naar het vakje direct links van de huidige positie. De pion mag niet naast het bord worden geplaatst.



In elk vierkant staat een positief geheel getal (niet in de tekening opgenomen). Als de speler zijn pion in een vierkant plaatst, dan wordt het daar genoteerde aantal punten aan zijn score toegevoegd.

Het spel is afgelopen als de pion is aangekomen in een vierkant met coördinaten $(0, 0)$.

lees verder

De bedoeling van deze opgave is een functie te maken die bij een gegeven speelbord bepaalt hoeveel punten er maximaal gescoord kunnen worden. Het speelbord wordt gerepresenteerd door een functie

```
pnt :: Integer -> Integer -> Integer
```

De expressie `(pnt r k)` is het aantal punten dat staat genoteerd in het vierkant met rijnummer `r` en kolomnummer `k`.

We zoeken een functie

```
maximumScore :: Integer -> Integer
```

met de betekenis: `(maximumScore r) =` het maximaal aantal te scoren punten als wordt begonnen in het vierkant met rijnummer `r` en kolomnummer `4`. Daartoe construeren we eerst de functie

```
maxScore :: Integer -> Integer -> Integer
```

met de betekenis: `(maxScore r k) =` het maximaal aantal te scoren punten als wordt begonnen in het vierkant met rijnummer `r` en kolomnummer `k`.

[6 pt] ☐ 1. Geef een Haskell-implementatie voor `maxScore`.

[3 pt] ☐ 2. Geef een Haskell-implementatie voor `maximumScore`

[5 pt] ☐ 3. We vragen een redelijke schatting voor de tijdcomplexiteit van de implementatie uit het vorige onderdeel. De complexiteitsfunctie is een polynoomfunctie van het rijnummer en we vragen alleen naar de graad van dit polynoom (dat is de hoogste macht). Licht je antwoord duidelijk toe.
HINT: bekijk eerst hoeveel rekenstappen er nodig zijn om `(maxScore n 0)` te bepalen. Daarna voor `(maxScore n 1)` etcetera.

[3 pt] ☐ 4. Leg uit wat we in het algemeen verstaan onder *dynamisch programmeren.*

[3 pt] ☐ 5. Leg uit waarom er mogelijk een efficiëntere implementatie bestaat voor `maxScore` die gebruik maakt van dynamisch programmeren.

[5 pt] ☐ 6. Geef een functie

```
maxScoreRij ::  Integer -> (Integer, Integer, Integer, Integer, Integer)
```

met de betekenis

```
maxScoreRij r  = ((maxScore r 0), (maxScore r 1), (maxScore r 2),
                        (maxScore r 3), (maxScore r 4))
```

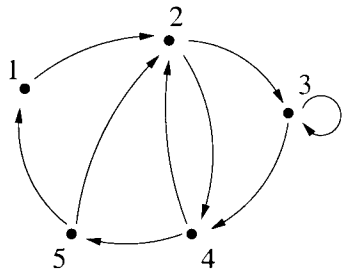
echter zonder `maxScore` zelf te gebruiken.

[3 pt] ☐ 7. Geef een nieuwe Haskell-implementatie voor `maximumScore`.

[4 pt] ☐ 8. Ook nu is de complexiteitsfunctie een polynoomfunctie. Geef de graad van dit polynoom. Beargumenteer je antwoord.

Opgave 2

Deze opgave gaat over grafen. Een graaf bestaat uit een verzameling punten of knopen en een verzameling takken of kanten. We nemen aan dat de knopen zijn gelabeld met gehele getallen; een tak wordt gerepresenteerd door een paar van gehele getallen: het label van het beginpunt van de tak en het label van het eindpunt van de tak. We hebben het hier dus over gerichte grafen.
De graaf representeren we door een lijst van takken. In deze lijst komen geen duplicaten voor. Voorbeeld:



De graaf hiernaast kan worden gerepresenteerd door de lijst

```
[ (1,2), (2,3), (3,3),
  (3,4), (4,2), (2,4),
  (4,5), (5,1), (5,2) ]
```

Merk op dat knopen die op geen enkele wijze zijn verbonden in deze representatie niet voorkomen.

[5 pt] □ 9. Onder de *uitgraad* van een knoop k in een graaf G verstaan we het aantal takken in G dat k als beginpunt heeft.
Geef een Haskell-functie `uitgraad` zo, dat

`(uitgraad k g) =` de uitgraad van `k` in de graaf `g`

Een graaf `h` heet een *deelgraaf* van de graaf `g` als de knopenverzameling van `h` een deelverzameling is van de knopenverzameling van `g` en bovendien de takkenverzameling van `h` een deelverzameling is van de takkenverzameling van `g`.

[5 pt] □ 10. Geef een Haskell-functie `isSubgraph` zo, dat

`(isSubgraph h g) =` `h` is een deelgraaf van `g`

HINT: ik heb hier een hulpfunctie gebruikt.

Een *kliek* in een graaf G is een verzameling L van knopen uit G met de eigenschap:

$$\text{voor elke } x \in L \text{ en } y \in L \text{ is } (x, y) \text{ een tak in } G$$

We ontwerpen nu een aantal functies om uiteindelijk een functie te maken die bepaalt of een gegeven verzameling L een kliek is in een graaf G.

[4 pt] □ 11. Geef een implementatie van de functie

```
connect :: Integer -> [Integer] -> [(Integer, Integer)]
```

die bij een knoop `k` en een lijst `lst` van knopen de lijst oplevert van alle takken van `k` naar elementen van `lst`. Bijvoorbeeld:

```
connect 3 [2, 3, 7, 5] = [(3,2), (3,3), (3,7), (3,5)]
```

[4 pt] □ 12. Geef een implementatie van de functie `product` die bij twee lijsten `p` en `q` van knopen de lijst oplevert van alle takken die ontstaan door elementen van `p` te verbinden met elementen van `q`.

[3 pt] □ 13. Geef een implementatie van de functie `complete` die bij een lijst `lst` van knopen de volledige graaf oplevert, dat wil zeggen de graaf waarin er een tak bestaat van iedere knoop van `lst` naar iedere knoop van `lst`.

[4 pt] □ 14. Geef een implementatie van de functie `isCliqueIn` met

`(isCliqueIn lst g) = lst` is een kliek in `g`

Bekijk nu het volgende beslissingsprobleem:

(CLIQUE)
Parameter: een graaf G en een positief geheel getal M
Gevraagd: heeft G een kliek van M punten?

[3 pt] □ 15. Leg uit wat we verstaan onder een beslissingsprobleem.

[3 pt] □ 16. Geef een `ja`-instantie van (**CLIQUE**).

[3 pt] □ 17. Geef een `nee`-instantie van (**CLIQUE**).

[4 pt] □ 18. Beargumenteer dat (**CLIQUE**) $\in$ **NP**.

Er valt zelfs te bewijzen dat (**CLIQUE**) $\in$ **NPC** door gebruik te maken van het feit dat knopenoverdekkingsprobleem (**VC**) $\in$ **NPC**.

[3 pt] □ 19. Leg uit wat we verstaan onder de klasse **NPC** .

[4 pt] □ 20. In welke richting moet de reductie gaan? Beargumenteer je antwoord.

Opgave 3

Gegeven is de volgende grammatica:

```
<S>  ::=   <A><S><B> | 'c'
<A>  ::=   'b'<A> | <A>'b' | 'a'
<B>  ::=   <A><A>
```

Het symbool `<S>` is het startsymbool.

[4 pt] ☐ 21. Beschrijf nauwkeurig de strings die kunnen worden afgeleid uit het hulpsymbool `<A>`.

[4 pt] ☐ 22. Teken een eindige automaat die van de invoer bepaalt of ze kan worden afgeleid uit het hulpsymbool `<B>`.

[4 pt] ☐ 23. Geef een afleidingsboom (parse tree) bij de string `bbabacbaaabbab`.

[4 pt] ☐ 24. Is de grammatica ambigu? Bewijs je bewering.

[5 pt] ☐ 25. Beschrijf nauwkeurig de strings die in deze grammatica afgeleid kunnen worden.

[6 pt] ☐ 26. Is er een eindige automaat die van de invoer nagaat of ze kan worden afgeleid uit het startsymbool `<S>`? Bewijs je bewering.

[4 pt] ☐ 27. Is er een Turingmachine die van de invoer nagaat of ze kan worden afgeleid uit het startsymbool `<S>`? Beargumenteer je bewering.

einde

totaal: 108 punten.