Tinpro01-7 Practicumopdracht 1

W. Oele

8 februari 2023

Opdracht 1a

Schrijf de functie

```
faca::Int->Int
faca x
```

Deze functie berekent de faculteit en maakt daarbij gebruik van pattern matching.

Opdracht 1b

Schrijf de functie

```
facb::Int->Int
facb x
```

Deze functie berekent de faculteit en maakt daarbij gebruik van guards.

Opdracht 2a

Schrijf de functie

```
nulpuntena::Double->Double->Double->[Double]
nulpuntena a b c
```

Deze functie berekent de nulpunten van een tweedegraads functie.

Opdracht 2b

Schrijf de functie

```
nulpuntenb::Double->Double->Double->[Double]
nulpuntenb a b c
```

Deze functie berekent de nulpunten van een tweedegraads functie en maakt daarbij gebruik van het `where` keyword en guards.

Opdracht 2c

We gooien met drie dobbelstenen. Schrijf een functie die een lijst teruggeeft met alle mogelijke worpen. Voor het representeren van één worp gebruiken we tuples, bijv. $(1, 2, 3)$. Schrijf de functie zodanig dat alleen de worpen, waarvan de som van de ogen een veelvoud is van vijf in de lijst worden opgenomen. Hoe groot is het aantal worpen?

Opdracht 2d

Herschrijf de functie uit opdracht 2c zodanig dat de lijst van tuples een veelvoud is van n.

Opdracht 3

Los de volgende puzzel op met Haskell:

We hebben 3 gehele getallen:

- Het eerste getal is 2 keer het verschil van het tweede en derde getal.
- Het tweede getal is het product van het eerste en derde getal.
- Het derde getal is de helft van de som van het eerste en tweede getal.

Wat zijn de waarden van het eerste, tweede en derde getal?

Opdracht 4a

We herinneren ons uit de module theoretische informatica de recurrente betrekkingen voor het vermenigvuldigen van twee positieve n-bits getallen.

- Schrijf de functie `mult::Integer->Integer->Integer`
- Deze functie is gebaseerd op de gedachte dat vermenigvuldigen hetzelfde is als herhaald optellen.
- Test je functie door steeds grotere getallen met elkaar te vermenigvuldigen. Ga door tot de Haskell interpreter het genoeg vindt en je getrakteerd wordt op een stack overflow. Zet de getallen die een stack overflow veroorzaken in het commentaar van je source.

Opdracht 4b

- Schrijf de functie `fastmult::Integer->Integer->Integer`
- Deze functie vermenigvuldigt twee positieve n-bit getallen en maakt daarbij handig gebruik van bitshifting.
- Voor het in Haskell kunnen uitvoeren van bitshifts heb je een library nodig: `Data.Bits`
- Bestudeer de `Data.Bits` library door de documentatie op hackage te lezen en te experimenteren met de aanwezige functies in de interpreter.
- Test je functie met dezelfde getallen als in opdracht 4a.

Opdracht 5a

We herinneren ons uit de module theoretische informatica de recurrente betrekkingen voor het machtsverheffen.

- Schrijf de functie `pow::Integer->Integer->Integer`. De functie rekent uit: $x^p, x > 0, p > 0$.
- Deze functie is gebaseerd op de gedachte dat machtsverheffen hetzelfde is als herhaald vermenigvuldigen.
- Test je functie met steeds grotere getallen. Ga door tot de Haskell interpreter het genoeg vindt en je getrakteerd wordt op een stack overflow. Zet de getallen die een stack overflow veroorzaken in het commentaar van je source.

Opdracht 5b

- Schrijf de functie `fastpow::Integer->Integer->Integer`. De functie rekent uit: x^p.
- Deze functie maakt handig gebruik van bitshifting.
- Test je functie met dezelfde getallen als in opdracht 4a.