FUNP taken– 2024-’25

Algemeen

Dit opleidingsonderdeel wordt voor de helft geëvalueerd via permanente evaluatie (= taken) en de helft op het examen. Op beide delen moet je minstens 8/20 halen om te kunnen slagen, maar ik reken erop dat dit voor niemand een probleem zal vormen.
In dit document worden de opgaven beschreven die tellen voor de permanente evaluatie. De opgaven zijn niet 100% precies geformuleerd zodat je zelf nog een invulling moet geven, net zoals je later in je carrière zal moeten doen omdat opdrachtgevers meestal ook niet op voorhand precies weten wat ze willen, en dat pas invullen naarmate het project vordert en de ingenieur informatica suggesties geeft. Zowel de code als de eigen invulling zijn deel van de mondelinge bespreking.

Omdat de deadline officieel tijdens de onderwijsperiode moet liggen, is de deadline 31 mei 2025.
De mondelinge bespreking van de taak gebeurt op de dag van het examen, maar je mag ook vroeger een afspraak maken.

Taak 1 – Hogere orde

Deze taak stond vermeld op de powerpoint bij de les van I/O en gaat over het aanpassen van de maanlander zodat die een functie meeneemt die de strategie bepaalt hoe de maanlander tegengas moet geven in de hoop dat die zo veilig landt.

Taak 2 – Type classes

De mate waarin twee elementen op elkaar lijken, wordt in veel algoritmes gebruikt, bijvoorbeeld in *recommender*systemen, in plagiaatcontrole, in het klasseren van elementen in gelijkaardige groepen bijvoorbeeld voor marktsegmentatie, …

Omdat het geen zin heeft een tekstdocument te vergelijken met een grafische figuur of een locatie met een recept, maar wel soortgelijke vragen kan stellen over verzamelingen van één type element, is deze opgave bijzonder geschikt om met ***type classes*** op te lossen.

Gevraagd is om

- Een **type class** op te stellen met minstens twee functies
    1. Een functie die twee elementen van hetzelfde type neemt en de mate van gelijkwaardigheid teruggeeft als een reëel getal in het bereik van 0 tot 1:
       0 is totaal géén gelijkenissen, en 1 volledig gelijkwaardige elementen.
        - Stel bijvoorbeeld dat je voor een String de mate waarin twee woorden dezelfde klinkers hebben als maatstaf neemt, dan hebben “ananas” en “komkommer” een gelijkwaardigheid van 0 (want geen gemeenschappelijke klinkers), “appel” en “dromen” 0.50 (want één van de twee klinkers gemeenschappelijk) en “peer” en “eten” 1.00 (want beide hebben ze 2x ‘e’)

    2. Een functie die twee elementen van hetzelfde type neemt en de afstand tussen twee elementen geeft. De afstand moet volgende eigenschappen hebben:
        - $d(a,b) \geq 0$
        - De afstand moet symmetrisch zijn: $d(a,b) = d(b,a)$
        - Het is niet noodzakelijk dat $d(x,y) = 0 \iff x = y$
        - Het moet wél een pseudometriek zijn: $x = y \implies d(x,y) = 0$
        - De afstand moet een verband hebben met de gelijkwaardigheid.

- Een manier om dat verband tussen afstand en gelijkwaardigheid te bekomen is bijvoorbeeld door de gelijkwaardigheid te berekenen door de afstand te delen door de maximale afstand tussen twee van die punten.
- Stel bijvoorbeeld dat je voor een String **afstand** definieert als het aantal verschillende letters, dan hebben zowel “appel” en “rappel” een afstand van 1 als “at” en “kat”. De **maximale afstand** tussen deze twee woorden is 6 respectievelijk 3 (het aantal letters van het langste woord).
  Als je de **gelijkwaardigheid** dan definieert als $\frac{maximale\ afstand - afstand}{maximale\ afstand}$ heb je altijd een getal tussen 0 en 1. De gelijkwaardig tussen “appel” en “rappel” is dan gelijk aan 5/6 of 0.833 en tussen “at” en “kat” gelijk aan 2/3 of 0.6666.

Beide functies moeten berekend kunnen worden uit de opgegeven waarde zelf.

- Definieer drie types en implementeer de type class voor elk van deze drie types.

- Daarnaast moet je de drie volgende functies implementeren.
  Ze staan opgesomd in stijgende moeilijkheidsgraad.

1. Geef alle elementen uit een lijst die minstens *x*% gelijkwaardig zijn aan een gegeven element. De functie heeft dus drie parameters: een lijst elementen, een getal x en een gegeven element.

2. Respecteert binnen een gegeven verzameling de functie afstand voor elk trio van elementen de driehoeksongelijkheid, nl. dat $d(a,b) + d\ (b,c) \geq= d(a,c)$?

3. Kan je in een gegeven verzameling van een element *a* tot een element *b* geraken met stappen in gelijkwaardigheid kleiner dan een opgegeven waarde *x*?
   - Voorbeeld: in [a,b,c,d] gelden volgende gelijkwaardigheden:

gelijkwaardigheid (s)	a	b	c	d
a	1	0.95	0.7	0.5
b	0.95	1	0.68	0.89
c	0.7	0.68	1	0.78
d	0.5	0.89	0.78	1

   Als je dan van *a* naar *d* wil gaan in stappen van maximaal 0.16, dan gaat dat:
   s(a,b) = 0.95 (stap = 0.05)
   s(b,d) = 0.89 (stap = 0.11)
   In stappen van 0.1 gaat het niet want enige stap vanuit a is naar b:
   s(a,b) = 0.95 (stap = 0.05)
   En van daaruit is elke gelijkwaardigheid kleiner dan 0.9.

Opgave (nog eens samengevat)
- Maak een type-class voor dit probleem.
- Implementeer deze type-class voor drie types.
- Implementeer de drie vermeldde functies.

Specifieke evaluatiecriteria (naast de algemene evaluatiecriteria op het einde van dit document)
- Complexiteit van de zelfgekozen types
- Originaliteit van de zelfgekozen types

Taak 3 – Homeswapping[1]

Context

Alice woont in Leuven maar werkt in Brussel. Bob daarentegen woont in Brussel maar werkt in Leuven. Ze kennen elkaar niet, maar ze hebben wel één ding gemeenschappelijk: ze zijn allebei het pendelen beu en zouden graag verhuizen naar een locatie dichter bij hun werk.
Zowel Alice en Bob kijken naar de immo-zoekertjes maar vinden niets naar hun zin. De oplossing ligt echter dichterbij dan ze zouden vermoeden. Het huis van Alice voldoet helemaal aan de wensen van Bob én omgekeerd. Ze zouden dus aan **homeswapping** kunnen doen.
Nu gaat dat niet om mensen pas verhuizen als er een woning vrij is die helemaal voldoet aan hun wensen. We zitten dus in een catch22-situatie: Bob kan niet verhuizen omdat het huis van Alice niet vrij is, en Alice kan niet verhuizen omdat het huis van Bob niet vrij.

Wat als we een nieuwe dienst zouden oprichten waarbij mensen hun verhuiswens bekend kunnen maken aan de wereld? In die verhuiswens zet je dan zowel informatie over de eigen woning als over de woning en locatie die men wil. Wanneer Alice en Bob dan allebei gebruik zouden maken, zouden ze elkaar kunnen vinden en zijn we weer een stap dichter bij de ideale wereld: Alice en Bob wonen veel dichter bij hun werkplek en verliezen minder tijd aan pendelen.

Opgave

Samengevat: bereken het maximaal aantal mensen dat een nieuwe woonst kan bekomen dankzij deze dienst.
Iets formeler:

- Zij W de verzameling van alle woningen ingeschreven in deze dienst, bv. W = {a, b, c, d, e, f}
- Voor elke woning $w \in W$, drukken we met de verzameling *c(W)* uit welke andere woonsten voldoen aan de wensen van de bewoners van woning w. De *c* staat voor *compatibeL*.
  Er geldt: $\forall w \in W: c(W) \subset W$
  - Per definitie ben je niet verplicht om te verhuizen als je deelneemt aan de dienst. Daarom geldt: $\forall w \in W: w \in c(W)$
- Een verhuisplan kent aan elke $w \in W$ één woonst toe. Zo'n verhuisplan duiden we aan met het symbool v. Hiervoor geldt: $\forall w \in W: v(w) \in c(W)$.
  - Het kan zijn dat er een verhuisplan is waarbij een aantal deelnemers niet verhuizen. Voor die mensen geldt dat *v(w) = w*.
- Een verhuisplan moet een bijectie zijn: twee verschillende personen kunnen niet naar eenzelfde woning verhuizen. Dit kunnen we uitdrukken als $\forall w, x \in W: v(w) = v(x) \Rightarrow w = x$.

Gevraagd: bereken het maximumaantal 'echte' verhuizingen (het aantal keren dat $v(w) \neq w$) voor de inschrijvingen in een concreet geval van deze dienst, waarbij dus zowel W als $\forall w \in W: c(W)$ gegeven zijn.

Bijkomend

- De invoer (W en $\forall w \in W: c(W)$) moet in een bestand beschreven staan en je oplossing moet dat bestand inlezen en de uitvoer afdrukken.
- Schrijf ook een Haskell-programma dat zo'n invoerbestand kan genereren.
- Hierbij moet je willekeurige waarden kunnen genereren. Installeer eerst de module System.Random door cabal install --lib random uit te voeren in de (power)shell.
  Daarna kan je met onderstaande code 10 gehele getallen, respectievelijk 10 letters afdrukken. Die code aanpassen voor nuttig gebruik is niet zo heel lastig.

[1] Deze opgave is geïnspireerd door een oude versie van de Vlaamse programmeerwedstrijd.

```
import System.Random

printInts = do
                g <- newStdGen
                let numbers = (randomRs (1, 10) g) :: [Integer]
                let n10 = take 10 numbers
                print n10

printChars = do
                g <- newStdGen
                let chars = (randomRs ('a', 'z') g)
                let c10 = take 10 chars
                print c10
```

Specifieke evaluatiecriteria (naast de algemene evaluatiecriteria op het einde van dit document)

- De mate waarin je in de generator voor invoerbestanden eigenschappen voor het invoerbestand kan genereren: bijvoorbeeld het aantal woningen in het bestand, maar ook bijvoorbeeld een onder- en bovengrens voor het aantal woningen in c(W), …

Algemene evaluatiecriteria:

- De mate waarin je de oplossing kan uitleggen en aanpassen, en de mate waarin je keuzes kan verantwoorden (**waarom** heb je dit/dat gedaan?).
- De mate waarin je alleen constructies gebruikt die we in de les behandeld hebben.
- Correctheid
- Functionele stijl

Vergeet niet dat bronvermelding belangrijk blijft! Als je code gevonden hebt (bv. op stackoverflow) of hebt laten genereren door genAI, geef dan ook aan welke bronnen en/of tools je gebruikt hebt.

Praktisch:

- De drie opgaven moeten in één bestand gestoken worden waarbij de bestandsnaam begint met je achternaam gevolgd door je voornaam. Upload dit bestand op Toledo.
- Het belangrijkste evaluatiecriterium is correctheid van de oplossing. Is dat in orde, dan haal je sowieso minstens 12/20, maar daarnaast wordt er ook naar de naamgeving van functies en parameters gekeken, naar een goed gebruik van de functionele stijl en de elegantie van de code (kort kan beter zijn dan langdradige code, maar wanneer de code zo kort is dat het overdreven cryptisch is en dat het niet meer duidelijk is wat er gebeurt (zie bv. de The International Obfuscated C Code Contest (https://www.ioccc.org/)) is het natuurlijk ook niet goed. Extra aanvullingen zijn ook een meerwaarde, maar geen must.
- Wanneer er errata zijn, mag je die melden. Die worden dan toegevoegd aan dit document en in de tekst op Toledo bij de uploadplek.

Kris Aerts
2 april 2025