UNIVERSITEIT VAN AMSTERDAM

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Tentamen

Programmeertalen

Bachelor Informatica jaar 1

Tentamen
Datum: 16 december 2015
Tijd: 09.00-12.00

Aantal pagina's: 5 (inclusief voorblad)
Aantal vragen: 22 + 1 bonusvraag
Maximaal aantal te behalen punten: 22 + 1 bonuspunt
Iedere vraag is precies één punt waard.

VOORDAT U BEGINT

- **Wacht** tot u de instructie krijgt het boekje te openen.
- Controleer of uw versie van het tentamen compleet is.
- Schrijf **uw naam en studentnummer en indien van toepassing versienummer** op **elk vel papier** dat u inlevert en **nummer de pagina's**.
- U dient uw **mobiele telefoon** uit te schakelen en te bewaren in uw jas of tas. Uw **jas en tas** moeten onder uw tafel liggen.
- **Toegestane hulpmiddelen**: kladpapier.

HUISHOUDELIJKE MEDEDELINGEN

- De eerste 30 minuten en de laatste 15 minuten mag u de zaal niet verlaten, ook niet voor het bezoeken van het toilet.
- Op verzoek van de examinator (of diens vertegenwoordiger) moet u zich kunnen legitimeren met een bewijs van inschrijving of een geldig legitimatiebewijs.
- Tijdens het tentamen is toiletbezoek niet toegestaan, tenzij de surveillant hier toestemming voor geeft.
- 15 minuten voor het eind wordt u gewaarschuwd dat het inlevertijdstip nadert.
- Vul indien van toepassing na afloop van het tentamen alstublieft het evaluatieformulier in.

Succes!

Dit tentamen begint met enkele algemene vragen en gaat dan in op alle behandelde programmeertalen. **Als er wordt gevraagd om uitleg, geef dan maximaal 3 regels uitleg.** Blijf kalm en probeer zo veel mogelijk vragen te beantwoorden. Heel veel succes!

Algemeen

1. Stel je voor dat je een backup procedure moet automatiseren. Welke programmeertaal kies je voor je programma en waarom?

2. In sommige opzichten lijken mailboxes van Erlang processen op channels in Go. Zijn mailboxes het meest vergelijkbaar met gebufferde, of met ongebufferde channels? Waaruit blijkt dat?

3. De openingszin uit Tolstojs Anna Karenina is wereldberoemd:

   > "Все счастливые семьи похожи друг на друга, каждая несчастливая семья несчастлива по-своему." (*Alle gelukkige gezinnen lijken op elkaar, elk ongelukkig gezin is ongelukkig op zijn eigen wijze.*)

   Vergelijk de return values van onderstaande functies en geef aan of een succesvolle uitvoering een 'gelukkig gezin' of een 'ongelukkig gezin' is ten aanzien van het Anna Karenina principe.

   Bash:

```
my_append() {
  if [[ ! -e $1 ]]; then
    return 1  # File does not exist
  elif [[ ! -w $1 ]]; then
    return 2  # File is not writable
  else
    echo "$2" >> "$1"
  fi
}
```

   Python:

```
def list_min(l):
    if not l:  # List is empty
        return None
    else:
        return min(l)
```

Bash

4. Wat is de 'return value' van een succesvolle aanroep van een functie, built-in of commando?

5. Wat is het algemene 'type' van variabelen in Bash?

6. Met welk commando open je de handleiding van een programma 'foo'?

7. Werkt onderstaande Bash code? Zo ja: wat doet het? Zo nee, waarom niet?

```
X="echo echo echo false"
while X=$($X); do
  echo mississippi
done
```

Prolog

8. Gegeven het volgende predikaat `foo/3`:

```
foo([],L,L).
foo([H|T],L2,[H|L3])  :-  foo(T,L2,L3).
```

   Welke built-in in Prolog komt overeen met dit predikaat?

9. Geeft het onderstaande statement `true` of `false`?

```
3+1 =:= 2+2.
```

10. Gegeven onderstaande code-fragment:

```
min(X,Y,X) :- X < Y.
min(X,Y,Y) :- X >= Y.
```

    (a) Geef een voorbeeld van een query waardoor een cut-operator gewenst is.

    (b) Neem het fragment over en plaats de cut-operator op de juiste plek.

Haskell

Voor onderstaande vragen kun je gebruik maken van de type-signatures van `foldl` en `foldr`:

```
foldl :: (b -> a -> b) -> b -> [a] -> b
foldr :: (a -> b -> b) -> b -> [a] -> b
```

11. De ingebouwde functie `or` geeft terug of een lijst van booleans minstens één `True` bevat. Definieer `or` doormiddel van foldr.

12. Wat is de uitvoer van het volgende stukje code?

```
foldl (\x y -> 2*x + 0.5 * y) 2 [4,1,3,3]
```

13. Waarom is het lastig om in een pure functionele taal een functie `get_user_count` te hebben die het aantal ingelogde gebruikers retourneert?

14. De built-in `elem` heeft de volgende type-signature:

```
elem :: Eq a => a -> [a] -> Bool
```

    Geef de type-signature van `uncurry elem` (ofwel de uitvoer van `:t uncurry elem`).

15. Wat is het verschil tussen het `Int` en `Integer` type in Haskell?

Python

16. Is onderstaande code geldig? Zo ja: wat is de uitvoer? Zo nee: waarom niet?

```
t = (1,2,3)
t[1] = 4
print sum(t)
```

17. Bekijk de onderstaande Python functie. Wat levert `hailstone(10)` op? En wat levert `hailstone(32)` op? Geef bij beide antwoorden aan of ze homogeen zijn.
(Hint: '//' is geheeltallige deling, zodat `5 // 2 == 2`)

```
def hailstone(n, *x):
    if n == 1:
        return x
    if n % 2 == 0:
        return hailstone(n // 2, n, *x)
    return hailstone(1 + 3 * n, (n,), x)
```

Go

18. Wat is de uitvoer van onderstaande stuk code?

```
package main

import "fmt"

func main() {
    var a [4]int
    var s []int
    a[2], a[1], a[0] = 57, 48, 37
    s = a[1:]
    s[1] = 1337
    for i := range a {
        fmt.Println(a[i])
    }
}
```

19. Waarom werkt onderstaand programma niet en wat moet eraan gebeuren om het te laten werken, ervan uitgaande dat het vooraf onbekend is hoeveel data er over het kanaal verstuurd wordt.

```
func new_producer(c chan string) {
        c <- "data"
}

func new_consumer(c chan string) {
        <-c
}

func main() {
        c := make(chan string)
        new_producer(c)
        new_consumer(c)
}
```

Erlang

20. Schrijf een list-comprehension die de Pythagoras-quadruplets genereert. (Een lijst met tuples $\{A, B, C, D\}$ waarvoor geldt dat $A^2 + B^2 + C^2 = D^2$, met $D \leq 10$).

21. Hoe representeert Erlang strings?

22. Gegeven een proces P, hoe verstuur je een string naar dat proces?

Bonus

23. Onderstaande code is geschreven in de taal C. In dit fragment zien we een globale variabele `g` en een functie `run` die als argument een verwijzing naar een functie, hier `cb` genoemd, accepteert. In dit geval is `cb` een functie die een type `int` argument heeft en een `int` retourneert. Volgt uit dit fragment dat C closures ondersteunt? Voorzie je antwoord van verdere toelichting.

    (a) Ja, dit fragment laat zien dat C closures ondersteunt.

    (b) Ja, dit fragment laat zien dat C geen closures ondersteunt.

    (c) Nee, op basis van dit fragment kan geen uitspraak worden gedaan over het ondersteunen van closures door C.

```
int g = 91;

int run(int (*cb)(int)) {
  return cb(g);
}
```