Examen 20 januari 2011

1. Unificatie (1pt (totaal 14ptn))

De Prolog built-in =/2 unificeert 2 termen. Schrijf unify/2 met dezelfde functionaliteit. Uiteraard maak je hierbij geen gebruik van =/2.

```
unify(X,X).
```

2. Steadfastness

Gegeven het predikaat max/3:

```
max(X, Y, X) :- !, X >= Y.
max(X, Y, Y).
```

a) Wat is het resultaat van de volgende query?- max(1, 2, R). false
b) Wat is het resultaat van de volgende query?- max(1, 2, 1). false
c) Indien de antwoorden verschillend zijn, herschrijf het predikaat zodat het steadfast wordt.

3. Bottom

Geef 3 implementaties voor f :: () -> () die we in Haskell van elkaar kunnen onderscheiden op basis van verschillende oproepen.

a) `f () = ()`
c) `f () = f ()`
c) `f _ = ()`

4. Clauses

Herschrijf de volgende clauses tot 1 clause:

```
p(z, z).
p(s(z), R) :- !, R = z.
p(s(X), Y) :- p(X, s(Y)).
```

```
q(A,B) :-
  ( A = z ->
    B = z
  ; A = s(z) ->
    B = z
  ; A = s(X) ->
    p(X,s(B))
  ).
```

5. Overbodige cuts (1pt)

Omcirkel de overbodige cuts.

```
q(a).
q(b).

p(1, X) :- !, q(X), !.
p(2, X) :- !, q(a), !.
```

```
p(1, X) :- !, q(X), !.
p(2, X) :- !, q(a), !.
```

6. DCGs (2pt)

Schrijf een DCG g/0 die gebalanceerde haakjes (voorgesteld door literals “open” en “close”) herkent.
Test op volgende queries:

```
phrase(g, [open, open, close, close], []).
        -> true.
phrase(g, [open, open, close, open, close, close], []).
        -> true.
phrase(g, [open, open, close], []).
        -> false.
```

```
g → [open],[close].
g → [open],g,[close].
g → g,g.
```

7. Type signatures (2pt)

Vul het meest algemene type in.

```
swap :: (a,b) -> (b,a)
swap (x,y) = (y,x)

inOrder :: Ord a => (a,a) -> Bool
inOrder (x,y) = y > x

loop :: a -> b
loop x = loop x

nil :: [a]
nil = []

optIn :: (Maybe a -> b) -> a -> b
optIn f x = f (Just x)
```

8. Vertalen (2pt)

Vertaal volgende list comprehension (k is een parameter) :
[x - y | x <- [1..5], y <- k, x < y]

tot een Prolog goal van de vorm
findall(Pattern, Goal, Result).

findall(Z, (member(X,[1,2,3,4,5]),member(Y,K),X < Y,Z is X - Y), Result).

9. Oneindige lijsten (2pt)

Schrijf een functie die de volgende oneindige lijsten genereert. Gebruik list comprehensions en/of de functie zelf (tying the knot).

a) Oneindige lijst van de oneven natuurlijke getallen [1, 3, 5, 7, …]
fa = 1 : map (+2) fa
fa = [x * 2 - 1 | x <- [1..]]

b) Oneindige lijst van oplopende sommen [0, a1, a1 + a2, a1 + a2 + a3, …] voor gegeven lijst l = [a1, a2, a3, …]
fb l = 0 : zipWith (+) l (fb l)

c) Oneindige lijst van triples (a, b, c) met a < b < c en $a^2 + b^2 = c^2$, in oplopende volgorde van c: [(3,4,5), (6,8,10), (5,12,13), …]
fc = [(a,b,c) | c <- [1..], b <- [1..c-1], a <- [1..b-1], a^2 + b^2 == c^2]

Examen 28 januari 2011

1. Functor

Herschrijf functor/3 zonder functor zelf te gebruiken.

2. Steadfastness

Gegeven het predikaat fac/2:

```
fac(0, 1) :- !.
fac(N, R) :-
 N1 is M - 1,
 fac(N1, R1),
 R is N * R1.
```

a) Wat is het resultaat van de volgende query?- fac(0, X).
b) Wat is het resultaat van de volgende query?- fac(0, 2).
c) Indien de antwoorden verschillend zijn, herschrijf het predikaat zodat het steadfast wordt.

3. Bottom

Gegeven:

```
f1 () = ()
f2 _ = ()
f3 () = f3 ()
```

Wat doen de volgende oproepen?

f1 ()	()
f1 undefined	undefined
f2 ()	()
f2 undefined	()
f3 ()	undefined
f3 undefined	undefined

4. Clauses

Herschrijf de volgende clauses tot 1 clause:

```
p([]).
p([X]).
p([X,Y|Xs]) :-
 X < Y,
 p([Y|Xs]).
```

5. Overbodige cuts (1pt)

Omcirkel de overbodige cuts, die altijd overbodig zijn.

```
q(a) :- !.
q(b) :- !.

p(1, X) :- q(X), !.
p(2, X) :- q(a), !.
```

6. DCGs (2pt)

Schrijf een DCG asbs/1 dat een string van n a's gevolgd door n b's herkent en de waarde van n teruggeeft.

phrase(asbs(N), [a,a,b,b],[]) geeft N=2
phrase(asbs(N),[a,b,a,b],[]) geeft false

7. Type signatures (2pt)

Vul het meest algemene type in.

```
func1::
nothing = Nothing

func2 ::
func2 x y
 | x < y = (x, y)
 | otherwise = (y, x)

func3 ::
func3 x = x + x

func4 ::
func4 f x y = f x y
```

8. Vertalen (2pt)

Vertaal volgende list comprehension (k is een parameter) :
[x*x | xs <- xss, x <- xs]

tot een Prolog goal van de vorm
findall(Pattern, Goal, Result).

9. Oneindige lijsten (2pt)

Schrijf een functie die de volgende oneindige lijsten genereert. Gebruik list comprehensions en/of de functie zelf (tying the knot).

a) Oneindige lijst van de natuurlijke getallen [2, 5, 8, 11, …]
fa =

b) Oneindige lijst van faculteitsgetallen [1, 1, 2, 6, 24, …]
fb l =

c) Oneindige lijst van triples (a, b, c) met a > b > c en $a^2 - b^2 = c^2$, in oplopende volgorde van c:
fc =