Philipps Universität Marburg

Deadline: 10.02.2014

Submit either at the exercise session on 10.02.2014 or in box 137 outside the toilets of Kern D.

Anonymous questionnaire about exercises
of the lecture "Programming languages and types"

- DO NOT write your name or student number (Matrikelnummer) on this page.
- DO NOT staple this page together with other pages of your homework submission.

The questionnaire is for statistics only and is in no way connected to the evaluation of your homework submission. By filling out this questionnaire, you declare to have answered as truthfully as your knowledge permits.

1. *Outside* the exercise session, how many hours did you spend doing the homework?

   _____ hours.

2. In your opinion, how many hours would it take an average student to do the homework?

   _____ hours.

3. How many lecture sessions did you attend this week?

   ☐ None. ☐ One session. ☐ Two sessions.

4. Did you attend the exercise session on 03.02.2014?

   ☐ Yes. ☐ No.

5. Please cross *all* statements that are true about you.

   ☐ I programmed in System F.
   ☐ I understand the meaning of System F types.
   ☐ I know how to structure an algorithm in functional programming style.
   ☐ I understand Velmans's predecessor function.

Philipps Universität Marburg

Deadline: 10.02.2014

Submit either at the exercise session on 10.02.2014 or in box 137 outside the toilets of Kern D.

Matrikelnummer:	
Name:	

Homework 13 of the lecture "Programming languages and types"

Solutions are available one week after deadline at
`https://www.uni-marburg.de/fb12/ps/teaching/ws13/plt-solutions/`

This homework is all about System F programming. You can obtain an interpreter of System F in the course repository:
`https://github.com/klauso/PLT2013/blob/master/exercises/ex13/F.jar?raw=true`

`F.jar` should be self-documenting when executed from the command line.
Definitions used in the exercises are available at:
`https://github.com/klauso/PLT2013/blob/master/exercises/ex13/examples.f`

1. (a) Given the church encoding of Booleans as

$$\begin{aligned} \textbf{type } \text{CBool} &= \forall\alpha.\ \alpha \to \alpha \to \alpha \\ \text{tru} &= \Lambda\alpha.\ \lambda x : \alpha.\ \lambda y : \alpha.\ x \\ \text{fls} &= \Lambda\alpha.\ \lambda x : \alpha.\ \lambda y : \alpha.\ y, \end{aligned}$$

write the logical conjunction and disjunction functions of Church Booleans in System F. Detailed description of the input-output behavior of these functions can be found between line 109 and 127 of `examples.f`.

and : CBool → CBool → CBool
and =

or : CBool → CBool → CBool
or =

(b) Given the church encoding of natural numbers as

type CNat = $\forall\alpha.\ (\alpha \to \alpha) \to \alpha \to \alpha$,

write a function "iszero" that returns the Church Boolean "tru" when applied to the Church numeral zero, and returns "fls" otherwise. The names "tru" and "fls" can be used in the definition.

iszero : CNat → CBool
iszero =

2. Behold "map", a polymorphic term that applies a function to every element in a list.

```
map : ∀α. ∀β. (α → β) → List α → List β
map = Λα. Λβ.
        λf : α → β. λxs : List α.
          isnil [α] xs [List β]
            (nil [β])
            (cons [β]
              (f (head [α] xs))
              (map [α] [β] f (tail [α] xs)))
```

Starting from the context

$$\begin{aligned}\Gamma = \quad & f : \alpha \to \beta,\\ & xs : \text{List } \alpha,\\ & \text{isnil} : \forall\delta.\ \text{List } \delta \to \text{CBool},\\ & \text{nil} : \forall\epsilon.\ \text{List } \epsilon,\\ & \text{cons} : \forall\zeta.\ \zeta \to \text{List } \zeta \to \text{List } \zeta,\\ & \text{head} : \forall\eta.\ \text{List } \eta \to \eta,\\ & \text{tail} : \forall\theta.\ \text{List } \theta \to \text{List } \theta,\\ & \text{map} : \forall\kappa.\ \forall\mu.\ (\kappa \to \mu) \to \text{List } \kappa \to \text{List } \mu,\end{aligned}$$

write down the types of the following subterms of the definition of "map".

$$\text{isnil } [\alpha] :$$

$$\text{isnil } [\alpha]\ xs :$$

$$\text{isnil } [\alpha]\ xs\ [\text{List } \beta] :$$

$$\text{nil } [\beta] :$$

$$\begin{aligned}&\text{isnil } [\alpha]\ xs\ [\text{List } \beta]\\ &\quad (\text{nil } [\beta]) \qquad :\end{aligned}$$

$$\text{cons } [\beta] :$$

$$\text{head } [\alpha]\ xs :$$

$$\text{cons } [\beta]\ (f\ (\text{head } [\alpha]\ xs)) :$$

$$\text{map } [\alpha]\ [\beta] :$$

$$\text{map } [\alpha]\ [\beta]\ f :$$

tail $[\alpha]$ xs :

map $[\alpha]$ $[\beta]$ f (tail $[\alpha]$ xs) :

cons $[\beta]$ (f (head $[\alpha]$ xs))
(map $[\alpha]$ $[\beta]$ f (tail $[\alpha]$ xs)) :

isnil $[\alpha]$ xs [List β]
(nil $[\beta]$)
(cons $[\beta]$
(f (head $[\alpha]$ xs))
(map $[\alpha]$ $[\beta]$ f (tail $[\alpha]$ xs))) :

3. (a) Write the polymorphic function "reverse" that reorders a list backwards. You may define helper functions.

reverse $[\mathbb{Z}]$ (cons $[\mathbb{Z}]$ 1 (cons $[\mathbb{Z}]$ 2 (cons $[\mathbb{Z}]$ 3 (nil $[\mathbb{Z}]$))))
$=$ cons $[\mathbb{Z}]$ 3 (cons $[\mathbb{Z}]$ 2 (cons $[\mathbb{Z}]$ 1 (nil $[\mathbb{Z}]$)))

reverse : $\forall\alpha$. List $\alpha \to$ List α
reverse $=$

(b) Implement any sorting algorithm in System F. Don't be surprised if the interpreter takes a long time to sort lists of 4 elements; call-by-name tends to be an inefficient evaluation strategy in practice. Don't be afraid to define helper functions, either.

$$\begin{aligned}\text{sort } [\mathbb{Z}] \;&\leq\; (\text{cons } [\mathbb{Z}]\ 4\ (\text{cons } [\mathbb{Z}]\ 1\ (\text{cons } [\mathbb{Z}]\ 3\ (\text{nil } [\mathbb{Z}]))))\\ &= \text{cons } [\mathbb{Z}]\ 1\ (\text{cons } [\mathbb{Z}]\ 3\ (\text{cons } [\mathbb{Z}]\ 4\ (\text{nil } [\mathbb{Z}])))\end{aligned}$$

$$\begin{aligned}&\text{sort} : \forall\alpha.\ (\alpha \to \alpha \to \text{CBool}) \to \text{List } \alpha \to \text{List } \alpha\\ &\text{sort} =\end{aligned}$$

4. (Optional) Pierce attributes the untyped term "vpred" to Barendregt, who attributes it to J. Velmans. It computes the predecessor of a Church numeral.

$$\begin{aligned}i &= \lambda x.\ x\\ k &= \lambda x.\ \lambda y.\ x\\ \text{vpred} &= \lambda n.\ \lambda s.\ \lambda z.\ n\ (\lambda p.\ \lambda q.\ q\ (p\ s))\ (k\ z)\ i\end{aligned}$$

(a) Insert type abstractions and type applications such that vpred has the type CNat $\to$ CNat in System F.

(b) Explain how "vpred" computes the predecessor of a Church numeral.