Fachbereich Mathematik und Informatik
Prof. Dr. K. Ostermann

Sebastian Erdweg, seba@informatik
Tillmann Rendel, rendel@informatik

Philipps Universität Marburg

November 23, 2009

Programming Languages and Types
Group Exercise 6

G6.1 Fibonacci in CPS

Define and CPS transform the Fibonacci function by hand.

G6.2 fold in CPS

Define and CPS transform fold by hand.

G6.3 MaybeCPS

Consider the following data type definition in Haskell syntax:

```
data MaybeCPS a
  = MaybeCPS (forall r.
               (a -> r) ->
               (() -> r) ->
               r)
```

1. Convince yourself that the given data type is essentially equivalent to the Maybe data type. How is `Nothing`, for instance, represented in MaybeCPS?

2. The definition of MaybeCPS gives rise to a monad. Define `return` and `bind` accordingly.

3. Evaluate the following expression in both, the Maybe and the MaybeCPS monad. Hereby, `nothing` denotes the representative of `Nothing` in the respective domain.

```
do v1 <- return 0
   v2 <- nothing
   v3 <- (v2 + 1)
   v4 <- return (v3 + 1)
   return v4
```

4. Compare the implementations of `return` and `bind` in the two monads. What differences do you see?