Fachbereich Mathematik und Informatik
Prof. Dr. K. Ostermann

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

Philipps Universität Marburg

October 19, 2009

Programming Languages and Types
Group Exercises G1

G1.1 Factorial in Scheme

Write a Scheme function `fact` which computes the factorial of a number.

```
> (fact 1)
1
> (fact 3)
6
> (fact 5)
120
```

G1.2 Lists in Scheme

Lists are predefined in Scheme, and can be used using the following functions. `'()` is the empty list. `cons` adds an element to the front of a list. `null` checks whether a list is empty. `car` returns the first element of a non-empty list. `cdr` returns everything but the first element of a non-empty list. `list` creates a list containing all of its arguments.

G1.2 a) Sum

Write a Scheme function `sum` which takes a list of numbers and returns the sum of all the numbers.

```
> (sum (list 5 17 11 10))
42
> (sum '())
0
```

G1.2 b) Product

Write a Scheme function `product` which takes a list of numbers and returns the product of all the numbers.

```
> (product (list 2 3 7))
42
> (product '())
1
```

please turn the page!

G1.3 Higher-Order Functions

Compare the code for `sum` and `product` from G1.2. Which parts of the code are identical, which are different? Write a function which abstracts over the different parts, and use it to define `sum` and `product`.

G1.4 Advanced Exercise

Use the function from G1.3 to define the identity function on lists `identity` and `map`.

```
> (identity '())
empty
> (identity (list 1 2 3))
(list 1 2 3)

> (map zero? (list 1 5 0 4 0))
(list false false true false true)
```