INFO020 - Langages de Programmation Evolués
Scheme Exercises 1

Roel Wuyts
roel.wuyts@ulb.ac.be

1 Evaluation of Expressions

Exercise 1: First guess the results of each of the following expressions. Then evaluate them and compare the results. Are they what you expected?

```
54
(+ 23 55)
(+ 23 44 99)
(+ 23 (- 55 44 33))
(* 2 (/ 8 4))
(/ 66 43)
(define a 3)
a
(/ 6 a)
(define b (+ a 1))
(+ a b (* a b))
+
```

Exercise 2: Continue with the following boolean expressions (so with a and b defined as in the previous exercise

```
(= 2 3)
(= 3 3)
(= a b)
(not (or (= 3 4) (= 5 6)))
(+ 2 (if (> a b) a b))
```

Exercise 3: Continue with the following conditional expressions (so with a and b defined as in the previous exercise

```
(if (= 1 1) "waaw" "brrr")
(if (> a b) a b)
(if (and (> b a) (< b (* a b))) b a)
(+ 2 (if (> a b) a b))
((if (< a b) + -) a b)
(cond
   ((= 1 1) "foo")
   ((= 2 2) "bar")
   ((= 3 3) "zork")
   (else "??que??"))
(* (cond  ((> a b) a)
```

```
((< a b) b)
(else -1))
(+ a 1))
```

Exercise 4: Express the following expression in Scheme and evaluate it: $\frac{\frac{12}{19}+\frac{5+9}{2}}{(10+11)*\frac{20}{3}}$

2 Procedures

Exercise 5: Write two functions celcius– >fahrenheit and fahrenheit– >celsius. The equation used is F = (C + 40) * 1, 8 - 40

Exercise 6: Write a procedure to calculate the Fibonacci numbers using the following recursive definition: $f(1) = 1, f(2) = 1, \forall n > 2 : f(n) = f(n-1) + f(n-2)$

Exercise 7: Write a procedure exp ***in Scheme for calculating*** b^n ***using the following recursive definition:*** $b^0 = 1, b^n = b * b^{n-1} if n \geq 1$

Exercise 8: ?Write a procedure fast-exp ***for calculating*** b^n ***that uses the following recursive definition:*** $b^0 = 1, b^n = b * b^{n-1}$ ***when n is odd,*** $b^n = (b^{n/2})^2$ ***when n is even. Then use the trace predicate to show that this algorithm is more efficient than the one from exercise 7. Optional: Can you make it even better?***

Exercise 9: Write a procedure to calculate the greatest common divisor (gcd) using the algorithm of Euclides: $pgcd(a,b) = aifa = b, pgcd(a,b) = pgcd(a-b,b)ifa > b, pgcd(a,b) = pgcd(a,b-a)ifb > a$***.***

Exercise 10: Write a recursive procedure displayn ***that takes two arguments, a character and a number. Using the procedure display, show the given character as many times as given by the number argument***

Exercise 11: Write a recursive function parametrized by a number that shows 4 squares of size number in such a way that those squares form a larger square themselves. For example:

```
> (squares 3)                > (squares 5)
          ******                     **********
          * ** *                     *    **    *
          ******                     *    **    *
          ******                     *    **    *
          * ** *                     **********
          ******                     **********
                                     *    **    *
                                     *    **    *
                                     *    **    *
                                     **********
```

Hint: use **displayn** ***from exercise 10.***

3 Higher-order procedures

Exercise 11: Write a procedure **(sum term a next b)** ***that takes two numbers (a and b) and two functions (term and next). The procedure sums all (term i), where i lies between a and b. The next i is found by applying the procedure next on the previous i. For example, we can use this to calculate all the squares of the first 10 integers as follows:***

```
(sum (lambda (x) (* x x)) 1 (lambda (x) (+ x 1)) 10)
```

Exercise 12: Write a procedure **(product term a next b)** ***analogous to exercise 11. This can then be used to, for example, calculate the product of all odd numbers between 1 and 10 as follows:***

```
(product (lambda (x) x) 1 (lambda (x) (+ x 2)) 10)
```

Exercise 13: Implementing the procedure **factorial** ***using the function*** **product** ***from exercise 12.***

Exercise 14: Write a procedure **(accumulate combiner null-value term a next b)** ***that abstracts away from the functions defined in exercises 11 and 12. Then rewrite*** **sum** ***and*** **product** ***in terms of*** **accumulate**

4 Lists

Exercise 15: First guess the results of each of the following expressions. Then evaluate them and compare the results. Are they what you expected?

```
()
(cons 1 2)
((car (cons (cons 1 2) (cons 3 4)))
(cons (cons (cons (cons 1 2) 3) 4) 5)
(cons 1 (cons 2 (cons 3 (cons 4 (cons 5 ())))))
```

```
(list 1 2 3 4 5)
(car (list 1 2 3 4 5))
(cdr (list 1 2 3 4 5))
(cadr (list 1 2 3 4 5))
(cadr (list 1 2 3 4 5))
```

Exercise 16: Write a procedure **member?** ***that checks whether a certain element is a member of a list.***

Exercise 17: Write a procedure **prefix?** ***that checks whether a certain list is the prefix of another list.***

Exercise 18: Write a procedure **my-inverse** ***to calculate the inverse of a list.***

Exercise 19: Write a procedure **my-length** ***to calculate the number of elements of a list.***

Exercise 20: Write a higher-order procedure **my-map** ***to construct a new list from a given list by applying a given procedure on each element. For example, we can construct the list of triples from a given list of integers as follows:***

```
(my-map (lambda (x) (* x 3)) '(1 2 3))
```