Principles of Programming, Spring 2006
Practice 2

Daejun Park, Heejong Lee
Programming Research Lab.@SNU

March 19, 2006

1. Define the factorial fuction called *fact* that takes a number as its argument and computes factorials. For examples,

   ```
   (fact 3)
   6

   (fact 0)
   1
   ```

2. Define a procedure called *combination* that takes two numbers as its argument, namely n, m, and computes ${}_nC_m$. For examples,

   ```
   (combination 4 2)
   6

   (combination 9 4)
   126
   ```

   Write two definitions of *combination* - one that uses above *fact* fuction and one that does not use multiplication($\times$) or division(/).[1]

3. Assume that you have three pegs and a set of disks, all of different diameters, with holes in them (so that they can slide onto the pegs). Start with all the disks on a single peg, in order of size (with the smallest on top). The object of the puzzle[2] is to move the pile of disks to a specified peg, by moving one disk at a time. A legal move consists of taking the top disk from any peg and putting it on either of the other two pegs; but a disk may never be placed on top of a disk that is smaller than itself.

   We will write a procedure *move-tower* that takes four arguments - the number of disks in the pile, the peg the disks are on, the peg the disks

[1] *Hint.* Consider *Pascal's triangle.*
[2] This puzzle is well known by 'Hanoi tower problem'

should be moved to, and the extra peg - and prints the sequence of moves. For example, consider moving three disks from peg 1 to peg 3 by evaluating *(move-tower 3 1 3 2)*. This should print:

```
move top disk from 1 to 3
move top disk from 1 to 2
move top disk from 3 to 2
move top disk from 1 to 3
move top disk from 2 to 1
move top disk from 2 to 3
move top disk from 1 to 3
```

You can use following procedure that takes two arguments - the peg the disks are on, the peg the disk should be moved to - and prints one step of moves.

```
(define (print-move from to)
  (newline)
  (display "move top disk from ") (display from)
  (display " to ") (display to))
```

4. Define a procedure called *sigma* that takes two numbers and a fuction as its arguments, namely a, b, f respectively, and computes the following.

$$\sum_{n=a}^{b} f(n) = f(a) + f(a+1) + \cdots + f(b)$$

For example,

```
(define (f n) (* n n))
(sigma 1 3 f)
14
```

5. Let f and g be two one-argument functions. The *composition* f after g is defined to be the function $x \mapsto f(g(x))$. Define a procedure *compose* that implements composition. For example,

```
(define (square x) (* x x))
(define (inc x) (+ x 1))
((compose square inc) 6)
49
```