```
define while = proc( test?, do, ans )
                letrec
                    loop = proc() if test? then { do; .loop() } else ans
                in
                    .loop()

define x = 0
define sum = 0
.while( <=?(x,10),
        {
            set sum = + ( sum, *( x, x ) );
            set x = add1( x )
        },
        sum
)
```

```
define pair = proc(x, y) proc(t) if t then y else x
define first = proc(p) .p(0) % p is assumed to be a pair
define second = proc(p) .p(1) % p is assumed to be a pair
define empty = 0          % give the "empty list" a name
define list = proc(x, xs) .pair(x, xs) % creates a nonempty list
define isEmpty = proc(xs) if xs then 0 else 1
define head = proc(xs) .first(xs) % the first element of the list
define tail  = proc(xs) .second(xs) % the (list of the) rest of the list

define at = proc(lst,i) if i then .at(.tail(lst),sub1(i)) else .head(lst)

define seq = proc(n) .list(n,.seq(add1(n)))
define natnos = .seq(1)
.at(natnos,0)
.at(natnos,1)
.at(natnos,100)
```