# CS 315

# PROGRAMMING LANGUAGES

# HW #3 REPORT

Berk Yıldız

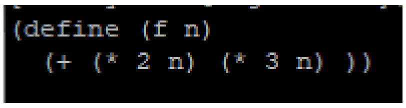
21502040

Section 1

I choose a linear function that takes a numeric value and returns a numeric value.

**f(n) = 2n + 3n**

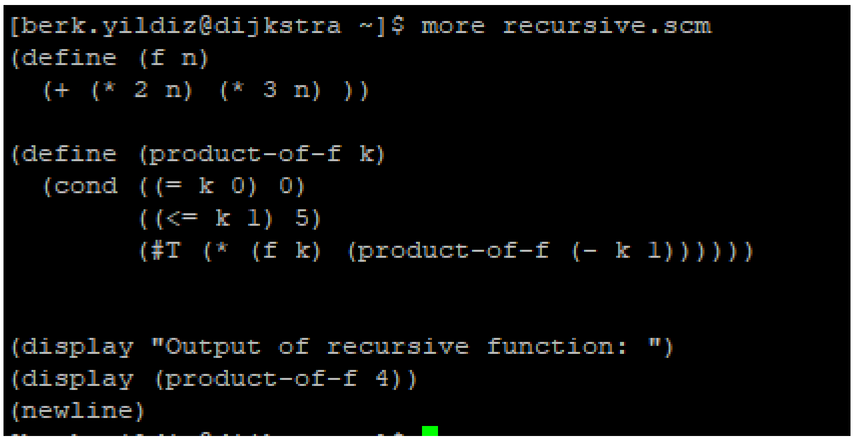
**Implementation of f(n):**



```
(define (f n)
  (+ (* 2 n) (* 3 n) ))
```

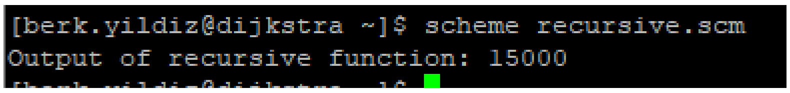
## Implementation of Recursive *product-of-f(k)*

**Code:**



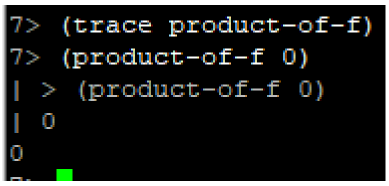
```
[berk.yildiz@dijkstra ~]$ more recursive.scm
(define (f n)
  (+ (* 2 n) (* 3 n) ))

(define (product-of-f k)
  (cond ((= k 0) 0)
        ((<= k 1) 5)
        (#T (* (f k) (product-of-f (- k 1))))))


(display "Output of recursive function: ")
(display (product-of-f 4))
(newline)
```

**Output for k=4:**



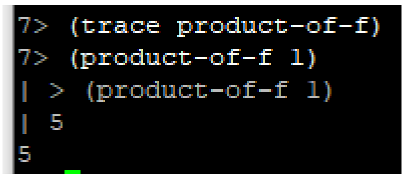
```
[berk.yildiz@dijkstra ~]$ scheme recursive.scm
Output of recursive function: 15000
```

**Trace for k=0:**



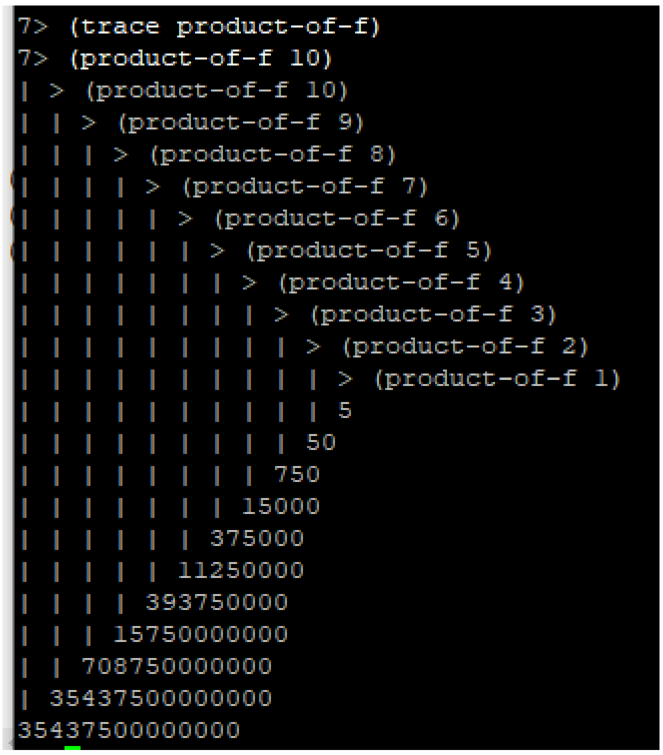
```
7> (trace product-of-f)
7> (product-of-f 0)
| > (product-of-f 0)
| 0
0
```

**Trace for k=1:**



```
7> (trace product-of-f)
7> (product-of-f 1)
| > (product-of-f 1)
| 5
5
```

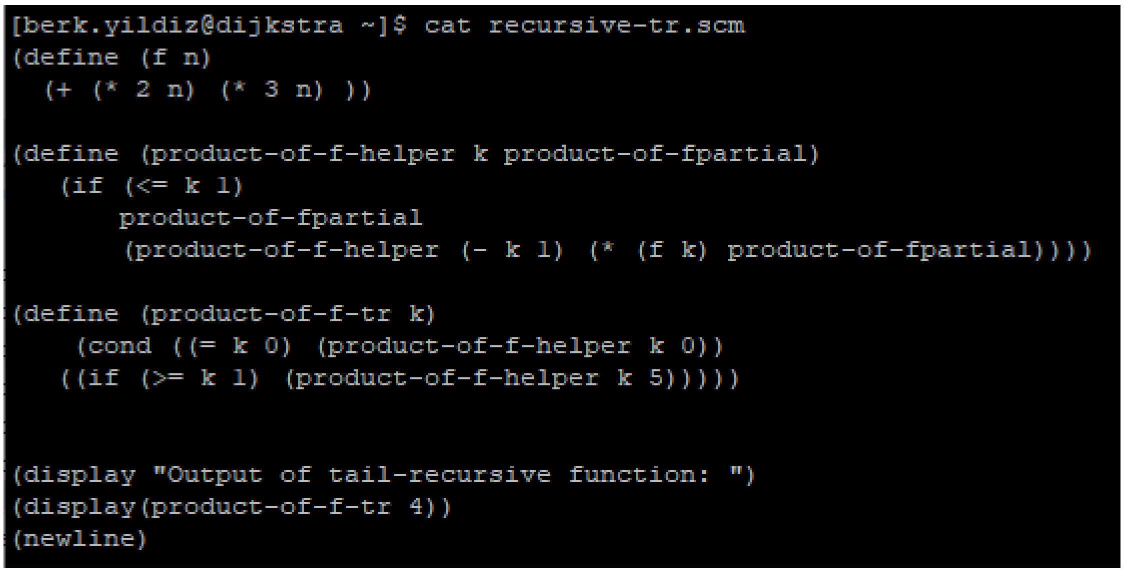
**Trace for k =10:**



```
7> (trace product-of-f)
7> (product-of-f 10)
| > (product-of-f 10)
| | > (product-of-f 9)
| | | > (product-of-f 8)
| | | | > (product-of-f 7)
| | | | | > (product-of-f 6)
| | | | | | > (product-of-f 5)
| | | | | | | > (product-of-f 4)
| | | | | | | | > (product-of-f 3)
| | | | | | | | | > (product-of-f 2)
| | | | | | | | | | > (product-of-f 1)
| | | | | | | | | | 5
| | | | | | | | | 50
| | | | | | | | 750
| | | | | | | 15000
| | | | | | 375000
| | | | | 11250000
| | | | 393750000
| | | 15750000000
| | 708750000000
| 35437500000000
35437500000000
```

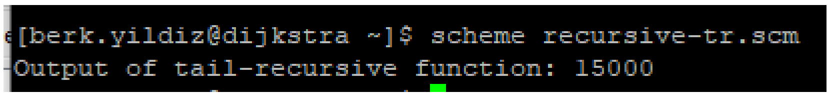
## Implementation of Tail-Recursive *product-of-f(k)*

**Code:**



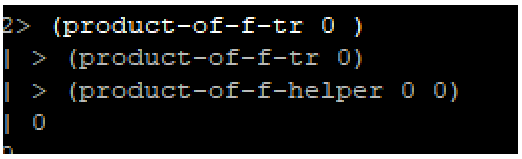
```
[berk.yildiz@dijkstra ~]$ cat recursive-tr.scm
(define (f n)
  (+ (* 2 n) (* 3 n) ))

(define (product-of-f-helper k product-of-fpartial)
   (if (<= k 1)
       product-of-fpartial
       (product-of-f-helper (- k 1) (* (f k) product-of-fpartial))))

(define (product-of-f-tr k)
    (cond ((= k 0) (product-of-f-helper k 0))
   ((if (>= k 1) (product-of-f-helper k 5)))))


(display "Output of tail-recursive function: ")
(display(product-of-f-tr 4))
(newline)
```

**Output for k=4:**



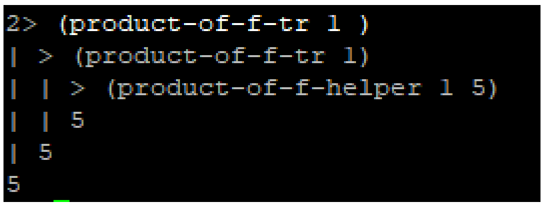
```
[berk.yildiz@dijkstra ~]$ scheme recursive-tr.scm
Output of tail-recursive function: 15000
```

## Trace for k=0:



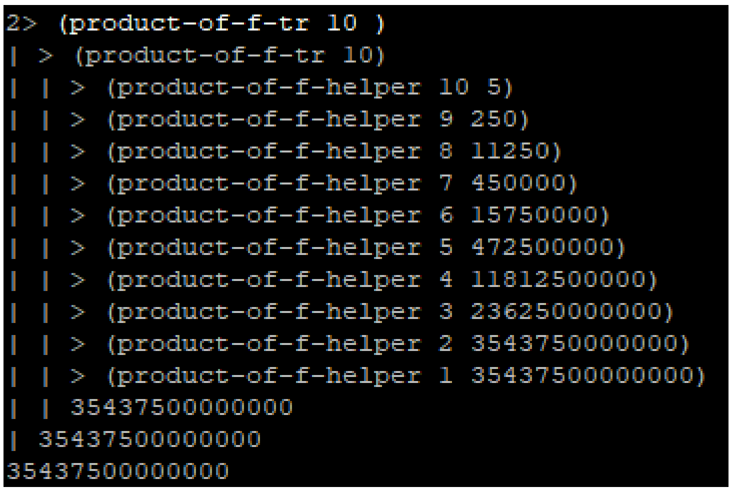
```
2> (product-of-f-tr 0 )
| > (product-of-f-tr 0)
| > (product-of-f-helper 0 0)
| 0
```

## Trace for k=1:



```
2> (product-of-f-tr 1 )
| > (product-of-f-tr 1)
| | > (product-of-f-helper 1 5)
| | 5
| 5
5
```

## Trace for k=10:



```
2> (product-of-f-tr 10 )
| > (product-of-f-tr 10)
| | > (product-of-f-helper 10 5)
| | > (product-of-f-helper 9 250)
| | > (product-of-f-helper 8 11250)
| | > (product-of-f-helper 7 450000)
| | > (product-of-f-helper 6 15750000)
| | > (product-of-f-helper 5 472500000)
| | > (product-of-f-helper 4 11812500000)
| | > (product-of-f-helper 3 236250000000)
| | > (product-of-f-helper 2 3543750000000)
| | > (product-of-f-helper 1 35437500000000)
| | 35437500000000
| 35437500000000
35437500000000
```