```
[dshin@acacia letrec]$ letrec /home/pl/hw07/tests/fact-of-5
Welcome to MzScheme v370 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ============================================================================
% fact. (value=120)

letrec fact(n)
  = if zero?(n)
    then 1
    else *(n,(fact -(n, 1)))
in (fact 5)

% ============================================================================
120
>
[dshin@acacia letrec]$ letrec /home/pl/hw07/tests/letrec-1
Welcome to MzScheme v370 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ============================================================================
% letrecs. (value=32)

letrec f(x)
  = -(x,1)
in (f 33)

% ============================================================================
32
>
[dshin@acacia letrec]$ letrec /home/pl/hw07/tests/letrec-2
Welcome to MzScheme v370 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ============================================================================
% letrecs. (value=8)

letrec f(x)
  = if zero?(x)
    then 0
    else -((f -(x,1)),-2)
in (f 4)

% ============================================================================
8
>
[dshin@acacia letrec]$ letrec /home/pl/hw07/tests/letrec-3
Welcome to MzScheme v370 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ============================================================================
% letrecs. (value=20)

let m=-5
in letrec f(x)
     = if zero?(x)
       then 0
       else -((f -(x,1)),m)
   in (f 4)

% ============================================================================
20
>
[dshin@acacia letrec]$ letrec /home/pl/hw07/tests/letrec-double
Welcome to MzScheme v370 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ============================================================================
% recursive double. (value=12)

letrec double(x)
  = if zero?(x)
    then 0
    else -((double -(x,1)),-(0, 2))
in (double 6)

```



```
% ============================================================================
12
>
[dshin@acacia letrec]$ letrec /home/pl/hw07/tests/letrec-sumto
Welcome to MzScheme v370 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ============================================================================
% recursive sumto. (value=55)

letrec sumto(n)
  = if zero?(n)
    then 0
    else -((sumto -(n,1)),-(0,n))
in (sumto 10)

% ============================================================================
55
>
[dshin@acacia letrec]$ letrec /home/pl/hw07/tests/proc-apply-1
Welcome to MzScheme v370 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ============================================================================
% proc application. (value=29)

(proc(x)-(x,1) 30)

% ============================================================================
29
>
[dshin@acacia letrec]$ letrec /home/pl/hw07/tests/proc-apply-2
Welcome to MzScheme v370 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ============================================================================
% proc application. (value=29)

let f=proc(x) -(x,1)
in (f 30)

% ============================================================================
29
>
[dshin@acacia letrec]$ letrec /home/pl/hw07/tests/proc-apply-3
Welcome to MzScheme v370 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ============================================================================
% proc can be applied twice. p75. (value=55)

let f=proc(x) -(x,11)
in (f (f 77))

% ============================================================================
55
>
[dshin@acacia letrec]$ letrec /home/pl/hw07/tests/proc-currying-1
Welcome to MzScheme v370 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ============================================================================
% multiple arguments with proc that returns proc. (value=-1)
% (This is called Currying.)

((proc(x)proc(y)-(x,y) 5) 6)

% ============================================================================
-1
>
[dshin@acacia letrec]$ letrec /home/pl/hw07/tests/proc-currying-2
Welcome to MzScheme v370 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ============================================================================
% multiple arguments with proc that returns proc. (value=-1)
% (This is called Currying.)

```

```
let f=proc(x)proc(y)-(x,y)
in ((f 5) 6)

% ========================================================================
-1
>
[dshin@acacia letrec]$ letrec /home/pl/hw07/tests/proc-currying-3
Welcome to MzScheme v370 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ========================================================================
% multiple arguments with proc that returns proc. (value=40)
% (This is called Currying.)

let plus=proc(x) proc(y) -(x,-(0,y))
in let minus=proc(x) proc(y) -(x,y)
   in ((minus ((plus 10) 20)) ((minus 40) 50))

% ========================================================================
40
>
[dshin@acacia letrec]$ letrec /home/pl/hw07/tests/proc-higher-1
Welcome to MzScheme v370 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ========================================================================
% proc's arg is proc. (value=29)
% (This is called higher-order function.)

(proc(f) (f 30)
 proc(x) -(x,1))

% ========================================================================
29
>
[dshin@acacia letrec]$ letrec /home/pl/hw07/tests/proc-higher-2
Welcome to MzScheme v370 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ========================================================================
% proc's arg is proc. p75. (value=55)
% (This is called higher-order function.)

(proc(f) (f (f 77))
 proc(x) -(x,11))

% ========================================================================
55
>
[dshin@acacia letrec]$ letrec /home/pl/hw07/tests/proc-in-let
Welcome to MzScheme v370 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ========================================================================
% procs in let is very useful. p76. (value=-100)

let x=200
in let f=proc(z) -(z,x)
   in let x=100
      in let g=proc(z) -(z,x)
         in -((f 1), (g 1))

% ========================================================================
-100
>
[dshin@acacia letrec]$
```