

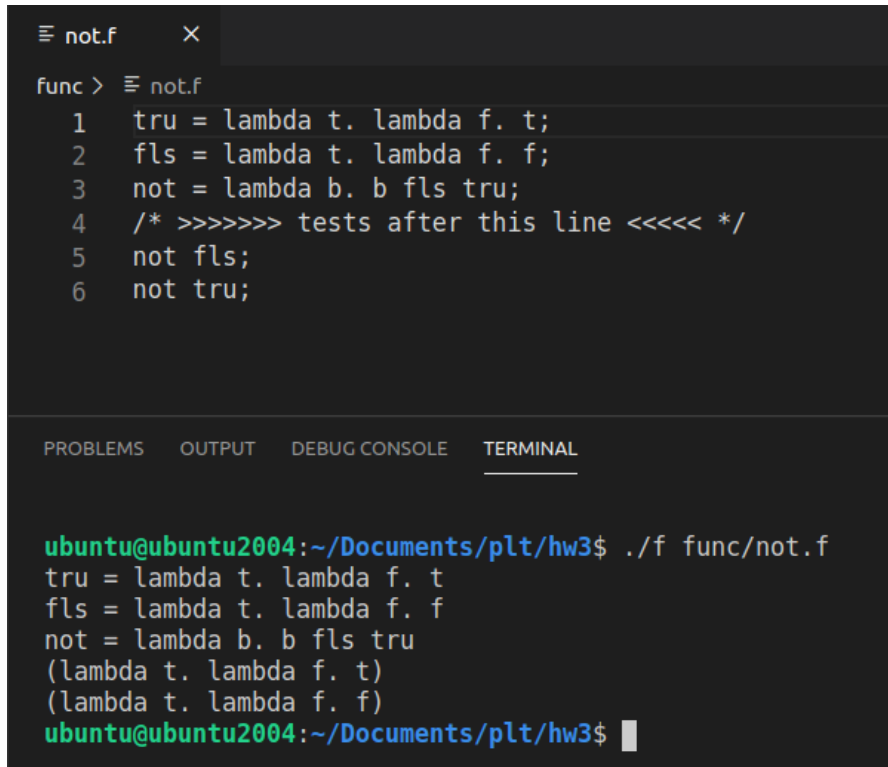
Exercise 1

$T = \lambda t. \lambda f. t$

$F = \lambda t. \lambda f. f$

NOT

$\text{NOT} = \lambda b. b F T$



The image shows a code editor window with a file named `not.f`. The editor contains the following code:

```
func > not.f
1  tru = lambda t. lambda f. t;
2  fls = lambda t. lambda f. f;
3  not = lambda b. b fls tru;
4  /* >>>>>> tests after this line <<<<< */
5  not fls;
6  not tru;
```

Below the editor, the terminal output shows the execution of the file:

```
ubuntu@ubuntu2004:~/Documents/plt/hw3$ ./f func/not.f
tru = lambda t. lambda f. t
fls = lambda t. lambda f. f
not = lambda b. b fls tru
(lambda t. lambda f. t)
(lambda t. lambda f. f)
ubuntu@ubuntu2004:~/Documents/plt/hw3$
```

NAND

$\text{NAND} = \lambda a. \lambda b. a \text{ F (NOT } b)$

```
func > ≡ nand.f
1  tru = lambda t. lambda f. t;
2  fls = lambda t. lambda f. f;
3  not = lambda b. b fls tru;
4  nand = lambda a. lambda b. a fls (not b);
5  /* >>>>>> tests after this line <<<<< */
6  nand tru tru;
7  nand fls tru;
8  nand tru fls;
9  nand fls fls;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: bash

```
ubuntu@ubuntu2004:~/Documents/plt/hw3$ ./f func/nand.f
tru = lambda t. lambda f. t
fls = lambda t. lambda f. f
not = lambda b. b fls tru
nand = lambda a. lambda b. a fls (not b)
(lambda t. lambda f. f)
(lambda t. lambda f. f)
(lambda t. lambda f. f)
(lambda t. lambda f. t)
ubuntu@ubuntu2004:~/Documents/plt/hw3$
```

XOR

$\text{XOR} = \lambda a. \lambda b. a \text{ (NOT } b) b$

```
func > ≡ xor.f
1  tru = lambda t. lambda f. t;
2  fls = lambda t. lambda f. f;
3  not = lambda b. b fls tru;
4  xor = lambda a. lambda b. a (not b) b;
5  /* >>>>>> tests after this line <<<<< */
6  xor tru fls;
7  xor fls tru;
8  xor tru tru;
9  xor fls fls;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: ba

```
ubuntu@ubuntu2004:~/Documents/plt/hw3$ ./f func/xor.f
tru = lambda t. lambda f. t
fls = lambda t. lambda f. f
not = lambda b. b fls tru
xor = lambda a. lambda b. a (not b) b
(lambda t. lambda f. t)
(lambda t. lambda f. t)
(lambda t. lambda f. f)
(lambda t. lambda f. f)
```

Exercise 2

$c0 = \lambda s. \lambda z. z$

$c1 = \lambda s. \lambda z. s z$

$c2 = \lambda s. \lambda z. s (s z)$

...

$scc = \lambda n. \lambda s. \lambda z. s (n s z)$

$pair = \lambda f. \lambda s. \lambda b. b f s$

$fst = \lambda p. p T$

$snd = \lambda p. P F$

$zz = pair c0 c0$

$step = \lambda p. pair (snd p) (scc (snd p))$

pred

pred = $\lambda n. \text{fst } (n \text{ step } z)$

```
func > pred.f
1 tru = lambda t. lambda f. t;
2 fls = lambda t. lambda f. f;
3 c_to_bool = lambda b. b true false;
4
5 c0 = lambda s. lambda z. z;
6 c1 = lambda s. lambda z. s (z);
7 c2 = lambda s. lambda z. s (s(z));
8 c3 = lambda s. lambda z. s (s(s(z)));
9 c4 = lambda s. lambda z. s (s(s(s(z))));
10 c5 = lambda s. lambda z. s (s(s(s(s(z)))));
11 scc = lambda n. lambda s. lambda z. s (n s z);
12 c_to_num = lambda cn. cn (lambda n. succ n);
13
14 pair = lambda f. lambda s. lambda b. b f s;
15 fst = lambda p. p tru;
16 snd = lambda p. p fls;
17
18 zz = pair c0 c0;
19 step = lambda p. pair (snd p) (scc (snd p));
20 predok = lambda n. fst (n step zz);
21 /* >>>>>> tests after this line <<<<<< */
22 c_to_num(predok c0);
23 c_to_num(predok c1);
24 c_to_num(predok c2);
25 c_to_num(predok c3);
26 c_to_num(predok c4);
27 c_to_num(predok c5);
28 iszero(c_to_num(predok c1));

ubuntu@ubuntu2004:~/Documents/pl1/hw3$ ./f func/pred.f
tru = lambda t. lambda f. t
fls = lambda t. lambda f. f
c_to_bool = lambda b. b true false
c0 = lambda s. lambda z. z
c1 = lambda s. lambda z. s z
c2 = lambda s. lambda z. s (s z)
c3 = lambda s. lambda z. s (s (s z))
c4 = lambda s. lambda z. s (s (s (s z)))
c5 = lambda s. lambda z. s (s (s (s (s z))))
scc = lambda n. lambda s. lambda z. s (n s z)
c_to_num = lambda cn. cn (lambda n. (succ n)) 0
pair = lambda f. lambda s. lambda b. b f s
fst = lambda p. p tru
snd = lambda p. p fls
zz = lambda b. b (lambda s. lambda z.z) (lambda s. lambda z.z)
step = lambda p. pair (snd p) (scc (snd p))
predok = lambda n. fst (n step zz)
0
0
1
2
3
4
true
ubuntu@ubuntu2004:~/Documents/pl1/hw3$
```

minus

minus = $\lambda a. \lambda b. b \text{ pred } a$

```
func > minus.f
1 tru = lambda t. lambda f. t;
2 fls = lambda t. lambda f. f;
3 c_to_bool = lambda b. b true false;
4
5 c0 = lambda s. lambda z. z;
6 c1 = lambda s. lambda z. s (z);
7 c2 = lambda s. lambda z. s (s(z));
8 c3 = lambda s. lambda z. s (s(s(z)));
9 c4 = lambda s. lambda z. s (s(s(s(z))));
10 c5 = lambda s. lambda z. s (s(s(s(s(z)))));
11 scc = lambda n. lambda s. lambda z. s (n s z);
12 c_to_num = lambda cn. cn (lambda n. succ n);
13
14 pair = lambda f. lambda s. lambda b. b f s;
15 fst = lambda p. p tru;
16 snd = lambda p. p fls;
17
18 zz = pair c0 c0;
19 step = lambda p. pair (snd p) (scc (snd p));
20 predok = lambda n. fst (n step zz);
21
22 minus = lambda a. lambda b. b predok a;
23 /* >>>>>> tests after this line <<<<<< */
24 c_to_num(minus c5 c3);
25 c_to_num(minus c2 c3);
26 c_to_num(minus c5 c1);
27 c_to_num(minus c4 c3);
28 c_to_num(minus c3 c1);

ubuntu@ubuntu2004:~/Documents/pl1/hw3$ ./f func/minus.f
tru = lambda t. lambda f. t
fls = lambda t. lambda f. f
c_to_bool = lambda b. b true false
c0 = lambda s. lambda z. z
c1 = lambda s. lambda z. s z
c2 = lambda s. lambda z. s (s z)
c3 = lambda s. lambda z. s (s (s z))
c4 = lambda s. lambda z. s (s (s (s z)))
c5 = lambda s. lambda z. s (s (s (s (s z))))
scc = lambda n. lambda s. lambda z. s (n s z)
c_to_num = lambda cn. cn (lambda n. (succ n)) 0
pair = lambda f. lambda s. lambda b. b f s
fst = lambda p. p tru
snd = lambda p. p fls
zz = lambda b. b (lambda s. lambda z.z) (lambda s. lambda z.z)
step = lambda p. pair (snd p) (scc (snd p))
predok = lambda n. fst (n step zz)
minus = lambda a. lambda b. b predok a
2
0
4
1
2
ubuntu@ubuntu2004:~/Documents/pl1/hw3$
```

isequal

AND = $\lambda a. \lambda b. a \text{ b } F$

iszero = $\lambda n. n (\lambda x. F) T$

leq = $\lambda a. \lambda b. \text{iszero (minus a b)}$

isequal = $\lambda a. \lambda b. \text{AND (leq a b) (leq b a)}$

```
isequal.f x
func > isequal.f
9  c3 = lambda s. lambda z. s (s(s(z)));
10 c4 = lambda s. lambda z. s (s(s(s(z))));
11 c5 = lambda s. lambda z. s (s(s(s(s(z)))));
12 scc = lambda n. lambda s. lambda z. s (n s z);
13
14 pair = lambda f. lambda s. lambda b. b f s;
15 fst = lambda p. p tru;
16 snd = lambda p. p fls;
17
18 zz = pair c0 c0;
19 step = lambda p. pair (snd p) (scc (snd p));
20 predok = lambda n. fst (n step zz);
21
22 minus = lambda a. lambda b. b predok a;
23
24 is0 = lambda n. n (lambda x. fls) tru;
25 leq = lambda a. lambda b. is0 (minus a b);
26 isequal = lambda a. lambda b. c_to_bool (and
27 /* >>>>>> tests after this line <<<<< */
28 isequal c1 c1;
29 isequal c2 c2;
30 isequal c3 c3;
31 isequal c4 c4;
32 isequal c5 c5;
33 isequal c1 c4;
34 isequal c3 c5;
35 isequal c2 c3;
```

```
ubuntu@ubuntu2004:~/Documents/pl1/hw3$ ./f func/minus.f
tru = lambda t. lambda f. t
fls = lambda t. lambda f. f
c_to_bool = lambda b. b true false
c0 = lambda s. lambda z. z
c1 = lambda s. lambda z. s z
c2 = lambda s. lambda z. s (s z)
c3 = lambda s. lambda z. s (s (s z))
c4 = lambda s. lambda z. s (s (s (s z)))
c5 = lambda s. lambda z. s (s (s (s (s z))))
scc = lambda n. lambda s. lambda z. s (n s z)
c_to_num = lambda cn. cn (lambda n. (succ n)) 0
pair = lambda f. lambda s. lambda b. b f s
fst = lambda p. p tru
snd = lambda p. p fls
zz = lambda b. b (lambda s. lambda z. z) (lambda s. lambda z. z)
step = lambda p. pair (snd p) (scc (snd p))
predok = lambda n. fst (n step zz)
minus = lambda a. lambda b. b predok a
2
0
4
1
2
ubuntu@ubuntu2004:~/Documents/pl1/hw3$
```

Exercise 3

Y-combinator

It is impossible to implement in call-by-value evaluation strategy.

Z-combinator

```

λ selection view Go Run Refresh Help
≡ fib.f  X
func > ≡ fib.f
1  fix = lambda f. (lambda x. f (lambda y. x x
2
3  plus = fix (lambda plus.
4      lambda a. lambda b.
5          if iszero a
6              then b
7              else succ (plus (pred a) b)
8  );
9
10 fib = fix (lambda fib.
11     lambda n.
12         if iszero n
13             then 0
14             else
15                 if iszero (pred n)
16                     then 1
17                     else plus (fib (pred n))
18 );
19 /* >>>>>>> tests after this line <<<<< */
20 fib 0;
21 fib 1;
22 fib 2;
23 fib 3;
24 fib 4;
25 fib 5;
26 fib 6;
27 fib 7;
28 fib 8;
29

```

```

lambda n'.
  if iszero n'
  then 0
  else if iszero (pred n')
    then 1
    else plus (fib (pred n'))
      (fib (pred (pred n'))))
  (lambda y'. x x y'))
y)
(pred n))
((lambda y.
  (lambda x.
    (lambda fib.
      lambda n'.
        if iszero n'
        then 0
        else if iszero (pred n')
          then 1
          else plus (fib (pred n'))
            (fib (pred (pred n'))))
    (lambda y'. x x y'))
  (lambda fib.
    lambda n'.
      if iszero n'
      then 0
      else if iszero (pred n')
        then 1
        else plus (fib (pred n'))
          (fib (pred (pred n'))))
    (lambda y'. x x y'))
  y)
(pred (pred n)))
0
1
1
2
3
5
8
13
21
ubuntu@ubuntu2004:~/Documents/plt/hw3$ history | tail -n 2
654 ./f func/fib.f
655 history | tail -n 2
ubuntu@ubuntu2004:~/Documents/plt/hw3$

```

Extra

real_to_c

This function converts regular to numbers to Church numerals, but for some reason it does not work with plus and minus.

```
func > real_to_c.f
17
18 plus = lambda a. lambda b. a scc b;
19
20 cn = lambda cn. lambda n. lambda s. lambda z
21 real_to_c = lambda n. cn cn n;
22 /* >>>>>> tests after this line <<<<<< */
23 c7 = real_to_c 7;
24 c11 = real_to_c 11;
25 c_to_real(c7);
26 c_to_real(predok(c7));
27 c_to_real(scc(c11));|
28
```

```

      lambda s'.
      lambda z'.
      if iszero n
      then z'
      else s' (cn' cn' (pred n) s' z'))
(lambda cn'.
  lambda n.
    lambda s'.
      lambda z'.
        if iszero n
        then z'
        else s' (cn' cn' (pred n) s' z'))
      (pred 7)
      s
      z)
c11 = lambda s.
  lambda z.
    if iszero 11
    then z
    else s
      ((lambda cn'.
        lambda n.
          lambda s'.
            lambda z'.
              if iszero n
              then z'
              else s' (cn' cn' (pred n) s' z'))
        (lambda cn'.
          lambda n.
            lambda s'.
              lambda z'.
                if iszero n
                then z'
                else s' (cn' cn' (pred n) s' z'))
          (pred 11)
          s
          z)
      7
      6
      12
ubuntu@ubuntu2004:~/Documents/plt/hw3$ history | tail -n 2
657 ./f func/real_to_c.f
658 history | tail -n 2
ubuntu@ubuntu2004:~/Documents/plt/hw3$
```