```
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/const-negative
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ========================================================================
% negative constant. (value=-27)

-27

% ========================================================================
|+ exp=-27
|| tenv=[x=int,v=int,i=int]
|- type=int
int
-27
>
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/const-positive
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ========================================================================
% positive constant. (value=12)

12

% ========================================================================
|+ exp=12
|| tenv=[x=int,v=int,i=int]
|- type=int
int
12
>
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/diff-nested-left
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ========================================================================
% nested diff. (-11)

-(-(44,33),22)

% ========================================================================
|+ exp=-(-(44,33),22)
|| tenv=[x=int,v=int,i=int]
||+ exp=-(44,33)
||| tenv=[x=int,v=int,i=int]
|||+ exp=44
|||| tenv=[x=int,v=int,i=int]
|||- type=int
|||+ exp=33
|||| tenv=[x=int,v=int,i=int]
|||- type=int
||- type=int
||+ exp=22
||| tenv=[x=int,v=int,i=int]
||- type=int
|- type=int
int
-11
>
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/diff-nested-right
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ========================================================================
% nested diff. (44)

-(55,-(22,11))

% ========================================================================
|+ exp=-(55,-(22,11))
|| tenv=[x=int,v=int,i=int]
||+ exp=55
||| tenv=[x=int,v=int,i=int]
```



```
||- type=int
||+ exp=-(22,11)
||| tenv=[x=int,v=int,i=int]
|||+ exp=22
|||| tenv=[x=int,v=int,i=int]
|||- type=int
|||+ exp=11
|||| tenv=[x=int,v=int,i=int]
|||- type=int
||- type=int
|- type=int
int
44
>
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/if-false-1
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ========================================================================
% if conditional. (value=4)

if zero?(1) then 3 else 4

% ========================================================================
|+ exp=if zero?(1) then 3 else 4
|| tenv=[x=int,v=int,i=int]
||+ exp=zero?(1)
||| tenv=[x=int,v=int,i=int]
|||+ exp=1
|||| tenv=[x=int,v=int,i=int]
|||- type=int
||- type=bool
||+ exp=3
||| tenv=[x=int,v=int,i=int]
||- type=int
||+ exp=4
||| tenv=[x=int,v=int,i=int]
||- type=int
|- type=int
int
4
>
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/if-false-type-error
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ========================================================================
% if conditional where false exp is not evaluated. (value=type-error)

if zero?(0) then 3 else foo

% ========================================================================
|+ exp=if zero?(0) then 3 else foo
|| tenv=[x=int,v=int,i=int]
||+ exp=zero?(0)
||| tenv=[x=int,v=int,i=int]
|||+ exp=0
|||| tenv=[x=int,v=int,i=int]
|||- type=int
||- type=bool
||+ exp=3
||| tenv=[x=int,v=int,i=int]
||- type=int
||+ exp=foo
||| tenv=[x=int,v=int,i=int]
apply-tenv: foo is undefined
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/tenv.scm:13
:23: (eopl:error (quote apply-tenv) "~a is undefined" search-sym)
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:1
5:4: (let-values (((val) (type-of-1 exp tenv))) (trace-check-exit val) val)
```

```
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:41:19: (let-values (((ty1) (type-of exp1 tenv)) ((ty2) (type-of exp2 tenv)) ((ty3) (type-of exp3 tenv))) (check-equal-type! ty1 (bool-type) exp1) (check-equal-type! ty2 ty3 exp) ty2)
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:15:4: (let-values (((val) (type-of-1 exp tenv))) (trace-check-exit val) val)
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/run.scm:7:30: (type->string (check string))
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/run.scm:7:15: (format "~a~n" (type->string (check string)))
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/run.scm:7:6: (display (format "~a~n" (type->string (check string))))
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/run.scm:5:4: (let-values (((string) (readfile file))) (display string) (display (format "~a~n" ((....) ....))) (set! trace-flag #f) (display (format (....) ....)) (newline))


 === context ===
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:13:2: type-of
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:20:2: type-of-1
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:13:2: type-of
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/run.scm:4:2: runfile

>
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/let
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ============================================================================
% let. (value=3)

let x=3
in x

% ============================================================================
|+ exp=let x=3 in x
|| tenv=[x=int,v=int,i=int]
||+ exp=3
||| tenv=[x=int,v=int,i=int]
||- type=int
||+ exp=x
||| tenv=[x=int,x=int,v=int,i=int]
||- type=int
|- type=int
int
3
>
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/letrec-1
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ============================================================================
% letrecs. (value=32)

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

% ============================================================================
|+ exp=letrec int f(x:int)=-(x,1) in (f 33)
|| tenv=[x=int,v=int,i=int]
||+ exp=-(x,1)
||| tenv=[x=int,f=(int->int),x=int,v=int,i=int]
|||+ exp=x
|||| tenv=[x=int,f=(int->int),x=int,v=int,i=int]
```



```
|||- type=int
|||+ exp=1
|||| tenv=[x=int,f=(int->int),x=int,v=int,i=int]
|||- type=int
||- type=int
||+ exp=(f 33)
||| tenv=[f=(int->int),x=int,v=int,i=int]
|||+ exp=f
|||| tenv=[f=(int->int),x=int,v=int,i=int]
|||- type=(int->int)
|||+ exp=33
|||| tenv=[f=(int->int),x=int,v=int,i=int]
|||- type=int
||- type=int
|- type=int
int
32
>
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/proc-1
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ============================================================================
% proc. (value=int->int)

proc(x:int) -(x,1)

% ============================================================================
|+ exp=proc(x:int)-(x,1)
|| tenv=[x=int,v=int,i=int]
||+ exp=-(x,1)
||| tenv=[x=int,x=int,v=int,i=int]
|||+ exp=x
|||| tenv=[x=int,x=int,v=int,i=int]
|||- type=int
|||+ exp=1
|||| tenv=[x=int,x=int,v=int,i=int]
|||- type=int
||- type=int
|- type=(int->int)
(int->int)
proc(x:int)-(x,1)[x=10,v=5,i=1]
>
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/proc-4
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ============================================================================
% proc. (value=(int->bool)->bool)

proc(f:(int->bool)) (f 3)

% ============================================================================
|+ exp=proc(f:(int->bool))(f 3)
|| tenv=[x=int,v=int,i=int]
||+ exp=(f 3)
||| tenv=[f=(int->bool),x=int,v=int,i=int]
|||+ exp=f
|||| tenv=[f=(int->bool),x=int,v=int,i=int]
|||- type=(int->bool)
|||+ exp=3
|||| tenv=[f=(int->bool),x=int,v=int,i=int]
|||- type=int
||- type=bool
|- type=((int->bool)->bool)
((int->bool)->bool)
proc(f:(int->bool))(f 3)[x=10,v=5,i=1]
>
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/proc-6
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
```

```
> % ========================================================================
% proc. (value=(int->((int->(int->bool))->(int->bool))))

proc (x:int) proc (f:(int->(int->bool))) (f x)

% ========================================================================
|+ exp=proc(x:int)proc(f:(int->(int->bool)))(f x)
|| tenv=[x=int,v=int,i=int]
||+ exp=proc(f:(int->(int->bool)))(f x)
||| tenv=[x=int,x=int,v=int,i=int]
|||+ exp=(f x)
|||| tenv=[f=(int->(int->bool)),x=int,x=int,v=int,i=int]
||||+ exp=f
||||| tenv=[f=(int->(int->bool)),x=int,x=int,v=int,i=int]
||||− type=(int->(int->bool))
||||+ exp=x
||||| tenv=[f=(int->(int->bool)),x=int,x=int,v=int,i=int]
||||− type=int
|||− type=(int->bool)
||− type=((int->(int->bool))->(int->bool))
|− type=(int->((int->(int->bool))->(int->bool)))
(int->((int->(int->bool))->(int->bool)))
proc(x:int)proc(f:(int->(int->bool)))(f x)[x=10,v=5,i=1]
>
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/proc-apply-1
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ========================================================================
% proc application. (value=29)

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

% ========================================================================
|+ exp=(proc(x:int)-(x,1) 30)
|| tenv=[x=int,v=int,i=int]
||+ exp=proc(x:int)-(x,1)
||| tenv=[x=int,v=int,i=int]
|||+ exp=-(x,1)
|||| tenv=[x=int,x=int,v=int,i=int]
||||+ exp=x
||||| tenv=[x=int,x=int,v=int,i=int]
||||− type=int
||||+ exp=1
||||| tenv=[x=int,x=int,v=int,i=int]
||||− type=int
|||− type=int
||− type=(int->int)
||+ exp=30
||| tenv=[x=int,v=int,i=int]
||− type=int
|− type=int
int
29
>
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/proc-currying-1
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ========================================================================
% multiple arguments with proc that returns proc. (value=-1)
% (This is called Currying.)

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

% ========================================================================
|+ exp=((proc(x:int)proc(y:int)-(x,y) 5) 6)
|| tenv=[x=int,v=int,i=int]
||+ exp=(proc(x:int)proc(y:int)-(x,y) 5)
||| tenv=[x=int,v=int,i=int]
```



```
|||+ exp=proc(x:int)proc(y:int)-(x,y)
|||| tenv=[x=int,v=int,i=int]
||||+ exp=proc(y:int)-(x,y)
||||| tenv=[x=int,x=int,v=int,i=int]
|||||+ exp=-(x,y)
|||||| tenv=[y=int,x=int,x=int,v=int,i=int]
||||||+ exp=x
|||||||| tenv=[y=int,x=int,x=int,v=int,i=int]
||||||− type=int
||||||+ exp=y
|||||||| tenv=[y=int,x=int,x=int,v=int,i=int]
||||||− type=int
|||||− type=int
||||− type=(int->int)
|||− type=(int->(int->int))
|||+ exp=5
|||| tenv=[x=int,v=int,i=int]
|||− type=int
||− type=(int->int)
||+ exp=6
||| tenv=[x=int,v=int,i=int]
||− type=int
|− type=int
int
−1
>
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/proc-type-error-1
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % ========================================================================
% proc. (value=type-error)

(proc(x:(int->int)) -(x,1) 30)

% ========================================================================
|+ exp=(proc(x:(int->int))-(x,1) 30)
|| tenv=[x=int,v=int,i=int]
||+ exp=proc(x:(int->int))-(x,1)
||| tenv=[x=int,v=int,i=int]
|||+ exp=-(x,1)
|||| tenv=[x=(int->int),x=int,v=int,i=int]
||||+ exp=x
||||| tenv=[x=(int->int),x=int,v=int,i=int]
||||− type=(int->int)
||||+ exp=1
||||| tenv=[x=(int->int),x=int,v=int,i=int]
||||− type=int
type-error: exp=x, expected type=(int->int), current type=int
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:85:4: (eopl:error (quote type-error) "exp=~a, expected type=~a, current type=~a" (exp->string exp) (type->string ty1) (type->string ty2))
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:29:21: (let-values (((ty1) (type-of exp1 tenv)) ((ty2) (type-of exp2 tenv))) (check-equal-type! ty1 (int-type) exp1) (check-equal-type! ty2 (int-type) exp2) (int-type))
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:15:4: (let-values (((val) (type-of-1 exp tenv))) (trace-check-exit val) val)
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:51:21: (let-values (((result-type) (type-of body (extend-tenv var ....)))) (proc-type var-type result-type))
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:15:4: (let-values (((val) (type-of-1 exp tenv))) (trace-check-exit val) val)
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:56:21: (let-values (((rator-type) (type-of rator tenv)) ((rand-type) (type-of rand tenv))) (let-values (((v) rator-type)) (if (not (....)) (error ....) (....))))
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:15:4: (let-values (((val) (type-of-1 exp tenv))) (trace-check-exit val) val)
```

```
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/run.scm:7:30: (type->string (check string))
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/run.scm:7:15: (format "~a~n" (type->string (check string)))
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/run.scm:7:6: (display (format "~a~n" (type->string (check string))))
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/run.scm:5:4: (let-values (((string) (readfile file))) (display string) (display (format "~a~n" ((....) ....))) (set! trace-flag #f) (display (format (....) ....)) (newline))


 === context ===
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:20:2: type-of-1
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:13:2: type-of
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:20:2: type-of-1
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:13:2: type-of
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:20:2: type-of-1
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:13:2: type-of
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/run.scm:4:2: runfile

>
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/proc-type-error-3
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % =====================================================================
% proc. (value=type-error)

proc (x:int) proc(f:(int->(int->bool))) (f zero?(x))

% ======================================================================
|+ exp=proc(x:int)proc(f:(int->(int->bool)))(f zero?(x))
|| tenv=[x=int,v=int,i=int]
||+ exp=proc(f:(int->(int->bool)))(f zero?(x))
||| tenv=[x=int,x=int,v=int,i=int]
|||+ exp=(f zero?(x))
|||| tenv=[f=(int->(int->bool)),x=int,x=int,v=int,i=int]
||||+ exp=f
||||| tenv=[f=(int->(int->bool)),x=int,x=int,v=int,i=int]
||||- type=(int->(int->bool))
||||+ exp=zero?(x)
||||| tenv=[f=(int->(int->bool)),x=int,x=int,v=int,i=int]
|||||+ exp=x
|||||| tenv=[f=(int->(int->bool)),x=int,x=int,v=int,i=int]
|||||- type=int
||||- type=bool
type-error: exp=zero?(x), expected type=int, current type=bool
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:85:4: (eopl:error (quote type-error) "exp=~a, expected type=~a, current type=~a" (exp->string exp) (type->string ty1) (type->string ty2))
[unknown source]: (let-values (((arg-type) (proc-type-accessor v 0)) ((result-type) (proc-type-accessor v 1))) (check-equal-type! arg-type rand-type rand) result-type)
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:15:4: (let-values (((val) (type-of-1 exp tenv))) (trace-check-exit val) val)
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:51:21: (let-values (((result-type) (type-of body (extend-tenv var ....)))) (proc-type var-type result-type))
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:15:4: (let-values (((val) (type-of-1 exp tenv))) (trace-check-exit val) val)
```



```
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:51:21: (let-values (((result-type) (type-of body (extend-tenv var ....)))) (proc-type var-type result-type))
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:15:4: (let-values (((val) (type-of-1 exp tenv))) (trace-check-exit val) val)
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/run.scm:7:30: (type->string (check string))
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/run.scm:7:15: (format "~a~n" (type->string (check string)))
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/run.scm:7:6: (display (format "~a~n" (type->string (check string))))
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/run.scm:5:4: (let-values (((string) (readfile file))) (display string) (display (format "~a~n" ((....) ....))) (set! trace-flag #f) (display (format (....) ....)) (newline))


 === context ===
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:20:2: type-of-1
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:13:2: type-of
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:20:2: type-of-1
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:13:2: type-of
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:20:2: type-of-1
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/check.scm:13:2: type-of
/home/dshin/2010-ea0011-pl/eopl3-programs/eopl3-interpreters/checked/run.scm:4:2: runfile

>
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/variable
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % =====================================================================
% variable, where env=[i=1,v=5,x=10]. (value=10)

x

% ======================================================================
|+ exp=x
|| tenv=[x=int,v=int,i=int]
|- type=int
int
10
>
dshin@acacia:checked$ checked -t /home/pl/hw09/tests/zero-true
Welcome to MzScheme v372 [3m], Copyright (c) 2004-2007 PLT Scheme Inc.
> % =====================================================================
% zero? (value=#t)

zero?(0)

% ======================================================================
|+ exp=zero?(0)
|| tenv=[x=int,v=int,i=int]
||+ exp=0
||| tenv=[x=int,v=int,i=int]
||- type=int
|- type=bool
bool
#t
>
dshin@acacia:checked$
```