```
Last login: Fri Feb 24 13:21:54 on ttys003
carbon:SamplePrograms$ cd Sec_01_1\:25pm/
carbon:Sec_01_1:25pm$ utop
                Welcome to utop version 1.14 (using OCaml version 4.01.0)!

Type #utop_help for help about using utop.

─( 18:00:00 )─< command 0 >──────────────────────────────────────{ counter: 0 }─
utop # #use "int_bool_expr.ml";;
File "int_bool_expr.ml", line 17, characters 13-18:
Error: Unbound type constructor value
─( 14:01:57 )─< command 1 >──────────────────────────────────────{ counter: 0 }─
utop # #use "int_bool_expr.ml";;
type expr =
    Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
  | Lt of expr * expr
  | Eq of expr * expr
  | And of expr * expr
  | Not of expr
  | If of expr * expr * expr
  | Let of string * expr * expr
  | Id of string
  | Value of value
and value = Int of int | Bool of bool
type environment = (string * value) list
File "int_bool_expr.ml", line 31, characters 4-16:
Error: This pattern matches values of type expr
       but a pattern was expected which matches values of type value * value
─( 14:03:09 )─< command 2 >──────────────────────────────────────{ counter: 0 }─
utop # #use "int_bool_expr.ml";;
type expr =
    Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
  | Lt of expr * expr
  | Eq of expr * expr
  | And of expr * expr
  | Not of expr
  | If of expr * expr * expr
  | Let of string * expr * expr
  | Id of string
  | Value of value
and value = Int of int | Bool of bool
type environment = (string * value) list
File "int_bool_expr.ml", line 25, characters 2-372:
```

```
Warning 8: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
(Mul (_, _)|Div (_, _)|Lt (_, _)|Eq (_, _)|And (_, _)|Not _|If (_, _, _)|
Let (_, _, _)|Id _)
val eval : environment -> expr -> value = <fun>
val e1 : expr = Add (Value (Int 1), Sub (Value (Int 10), Value (Int 3)))
-( 14:03:41 )-< command 3 >-------------------------------------{ counter: 0 }-
utop # eval [] e1 ;;
- : value = Int 8
-( 14:04:56 )-< command 4 >-------------------------------------{ counter: 0 }-
utop # #use "int_bool_expr.ml";;
type expr =
    Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
  | Lt of expr * expr
  | Eq of expr * expr
  | And of expr * expr
  | Not of expr
  | If of expr * expr * expr
  | Let of string * expr * expr
  | Id of string
  | Value of value
and value = Int of int | Bool of bool
type environment = (string * value) list
File "int_bool_expr.ml", line 29, characters 26-35:
Error: This expression has type int but an expression was expected of type
         value
-( 14:05:01 )-< command 5 >-------------------------------------{ counter: 0 }-
utop # eval [] (Add (Lt (Value (Int 5), Value (Int 3)), Value (Int 4)));;
Exception: Match_failure ("int_bool_expr.ml", 25, 2).
-( 14:05:55 )-< command 6 >-------------------------------------{ counter: 0 }-
utop # #use "int_bool_expr.ml";;
type expr =
    Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
  | Lt of expr * expr
  | Eq of expr * expr
  | And of expr * expr
  | Not of expr
  | If of expr * expr * expr
  | Let of string * expr * expr
  | Id of string
  | Value of value
and value = Int of int | Bool of bool
type environment = (string * value) list
File "int_bool_expr.ml", line 25, characters 2-543:
Warning 8: this pattern-matching is not exhaustive.
```

```
Here is an example of a value that is not matched:
(Mul (_, _)|Div (_, _)|Eq (_, _)|And (_, _)|Not _|If (_, _, _)|Let (_, _, _)|
Id _)
val eval : environment -> expr -> value = <fun>
val e1 : expr = Add (Value (Int 1), Sub (Value (Int 10), Value (Int 3)))
─( 14:08:43 )─< command 7 >──────────────────────────────────────{ counter: 0 }─
utop # eval [] (Add (Lt (Value (Int 5), Value (Int 3)), Value (Int 4)));;
Exception: Failure "incompatible type on Add".
─( 14:09:08 )─< command 8 >──────────────────────────────────────{ counter: 0 }─
utop #
```

Add	And	Arg	Arith_status	Array	ArrayLabels	Assert_failure	Big_int	Bigarray	Boo