```
Last login: Mon Feb 13 13:13:16 on ttys007
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 "binary_tree.ml";;
type 'a tree = Leaf of 'a | Fork of 'a * 'a tree * 'a tree
val t1 : int tree = Leaf 5
val t2 : int tree = Fork (3, Leaf 3, Fork (2, Leaf 5, Leaf 5))
─( 13:30:16 )─< command 1 >──────────────────────────────────{ counter: 0 }─
utop # #use "binary_tree.ml";;
type 'a tree = Leaf of 'a | Fork of 'a * 'a tree * 'a tree
val t1 : int tree = Leaf 5
val t2 : int tree = Fork (3, Leaf 3, Fork (2, Leaf 5, Leaf 5))
val t3 : string tree = Fork ("Hello", Leaf "World", Leaf "!")
─( 13:30:19 )─< command 2 >──────────────────────────────────{ counter: 0 }─
utop # [1;2;3] ;;
- : int list = [1; 2; 3]
─( 13:30:53 )─< command 3 >──────────────────────────────────{ counter: 0 }─
utop # ['a'; 'b'] ;;
- : char list = ['a'; 'b']
─( 13:31:04 )─< command 4 >──────────────────────────────────{ counter: 0 }─
utop # #use "binary_tree.ml";;
type inttree = ILeaf of int | IFork of int * inttree * inttree
type 'a tree = Leaf of 'a | Fork of 'a * 'a tree * 'a tree
val t1 : int tree = Leaf 5
val t2 : int tree = Fork (3, Leaf 3, Fork (2, Leaf 5, Leaf 5))
val t3 : string tree = Fork ("Hello", Leaf "World", Leaf "!")
─( 13:31:11 )─< command 5 >──────────────────────────────────{ counter: 0 }─
utop # #use "binary_tree.ml";;
type inttree = ILeaf of int | IFork of int * inttree * inttree
type 'a tree = Leaf of 'a | Fork of 'a * 'a tree * 'a tree
val t1 : int tree = Leaf 5
val t2 : int tree = Fork (3, Leaf 3, Fork (2, Leaf 5, Leaf 5))
val t3 : string tree = Fork ("Hello", Leaf "World", Leaf "!")
val size : 'a tree -> int = <fun>
─( 13:39:31 )─< command 6 >──────────────────────────────────{ counter: 0 }─
utop # size t1 ;;
- : int = 1
─( 13:42:41 )─< command 7 >──────────────────────────────────{ counter: 0 }─
utop # size t2 ;;
- : int = 5
─( 13:42:46 )─< command 8 >──────────────────────────────────{ counter: 0 }─
utop # #use "binary_tree.ml";;
type inttree = ILeaf of int | IFork of int * inttree * inttree
type 'a tree = Leaf of 'a | Fork of 'a * 'a tree * 'a tree
```

```
val t1 : int tree = Leaf 5
val t2 : int tree = Fork (3, Leaf 3, Fork (2, Leaf 5, Leaf 5))
val t3 : string tree = Fork ("Hello", Leaf "World", Leaf "!")
val size : 'a tree -> int = <fun>
val sum : int tree -> int = <fun>
─( 13:42:49 )─< command 9 >──────────────────────────────────────{ counter: 0 }─
utop # sum t2 ;;
- : int = 18
─( 13:43:52 )─< command 10 >─────────────────────────────────────{ counter: 0 }─
utop # sum t3 ;;
Error: This expression has type string tree
       but an expression was expected of type int tree
       Type string is not compatible with type int
─( 13:44:19 )─< command 11 >─────────────────────────────────────{ counter: 0 }─
utop # #use "binary_tree.ml";;
type inttree = ILeaf of int | IFork of int * inttree * inttree
type 'a tree = Leaf of 'a | Fork of 'a * 'a tree * 'a tree
val t1 : int tree = Leaf 5
val t2 : int tree = Fork (3, Leaf 3, Fork (2, Leaf 5, Leaf 5))
val t3 : string tree = Fork ("Hello", Leaf "World", Leaf "!")
val tmap : ('a -> 'b) -> 'a tree -> 'b tree = <fun>
val size : 'a tree -> int = <fun>
val size' : 'a tree -> int = <fun>
val sum : int tree -> int = <fun>
─( 13:44:23 )─< command 12 >─────────────────────────────────────{ counter: 0 }─
utop # tmap (fun x -> x + 1) t2 ;;
- : int tree = Fork (4, Leaf 4, Fork (3, Leaf 6, Leaf 6))
─( 13:51:49 )─< command 13 >─────────────────────────────────────{ counter: 0 }─
utop # tmap String.length t3 ;;
- : int tree = Fork (5, Leaf 5, Leaf 1)
─( 13:51:49 )─< command 14 >─────────────────────────────────────{ counter: 0 }─
utop # #use "binary_tree.ml";;
type inttree = ILeaf of int | IFork of int * inttree * inttree
type 'a tree = Leaf of 'a | Fork of 'a * 'a tree * 'a tree
val t1 : int tree = Leaf 5
val t2 : int tree = Fork (3, Leaf 3, Fork (2, Leaf 5, Leaf 5))
val t3 : string tree = Fork ("Hello", Leaf "World", Leaf "!")
val tmap : ('a -> 'b) -> 'a tree -> 'b tree = <fun>
val tfold : ('a -> 'b -> 'b) -> 'b -> 'a tree -> 'b = <fun>
val size : 'a tree -> int = <fun>
val size' : 'a tree -> int = <fun>
val sum : int tree -> int = <fun>
─( 13:52:32 )─< command 15 >─────────────────────────────────────{ counter: 0 }─
utop # tfold (+) 0 t2 ;;
- : int = 18
─( 14:00:09 )─< command 16 >─────────────────────────────────────{ counter: 0 }─
utop # List.fold_right ;;
- : ('a -> 'b -> 'b) -> 'a list -> 'b -> 'b = <fun>
─( 14:00:24 )─< command 17 >─────────────────────────────────────{ counter: 0 }─
utop # List.fold_right (fun h t -> h :: t) [1;2;3;4] [] ;;
- : int list = [1; 2; 3; 4]
```

```
─( 14:01:26 )─< command 18 >──────────────────────────────────{ counter: 0 }─
utop # List.fold_right (fun h t -> (h + 1) :: t) [1;2;3;4] [] ;;
- : int list = [2; 3; 4; 5]
─( 14:01:52 )─< command 19 >──────────────────────────────────{ counter: 0 }─
utop # List.fold_right (fun h t -> (h + 1) :: t) (1::2::3::4::[]) [] ;;
- : int list = [2; 3; 4; 5]
─( 14:02:20 )─< command 20 >──────────────────────────────────{ counter: 0 }─
utop # List.fold_right (+) (1::2::3::4::[]) [] ;;
Error: This expression has type 'a list but an expression was expected of type
         int
─( 14:10:40 )─< command 21 >──────────────────────────────────{ counter: 0 }─
utop # List.fold_right (+) (1::2::3::4::[]) 0 ;;
- : int = 10
─( 14:10:40 )─< command 22 >──────────────────────────────────{ counter: 0 }─
utop # #use "binary_tree.ml";;
type inttree = ILeaf of int | IFork of int * inttree * inttree
type 'a tree = Leaf of 'a | Fork of 'a * 'a tree * 'a tree
val t1 : int tree = Leaf 5
val t2 : int tree = Fork (3, Leaf 3, Fork (2, Leaf 5, Leaf 5))
val t3 : string tree = Fork ("Hello", Leaf "World", Leaf "!")
val tfold : ('a -> 'b) -> ('a -> 'b -> 'b -> 'b) -> 'a tree -> 'b = <fun>
val tmap : ('a -> 'b) -> 'a tree -> 'b tree = <fun>
val tfold' : ('a -> 'b -> 'b) -> 'b -> 'a tree -> 'b = <fun>
val size : 'a tree -> int = <fun>
val size' : 'a tree -> int = <fun>
val sum : int tree -> int = <fun>
─( 14:11:00 )─< command 23 >──────────────────────────────────{ counter: 0 }─
utop # tfold (fun x -> x) (fun a b c -> a + b + c)   t2 ;;
- : int = 18
─( 14:12:54 )─< command 24 >──────────────────────────────────{ counter: 0 }─
utop # tfold (fun x -> Left x) (fun a b c -> Fork (a, b, c))   t2 ;;
Error: Unbound constructor Left
─( 14:15:05 )─< command 25 >──────────────────────────────────{ counter: 0 }─
utop # tfold (fun x -> Leaf x) (fun a b c -> Fork (a, b, c))   t2 ;;
- : int tree = Fork (3, Leaf 3, Fork (2, Leaf 5, Leaf 5))
─( 14:15:25 )─< command 26 >──────────────────────────────────{ counter: 0 }─
utop #
```

Arg	Arith_status	Array	ArrayLabels	Assert_failure	Big_int	Bigarray	Buffer	Call