```
Last login: Mon Feb 13 15:43:32 on ttys013
carbon:SamplePrograms$ cd Sec_10
-bash: cd: Sec_10: No such file or directory
carbon:SamplePrograms$ cd Sec_10_3\:35pm/
carbon:Sec_10_3:35pm$ 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
File "binary_tree.ml", line 22, characters 9-41:
Error: The constructor Fork expects 3 argument(s),
       but is applied here to 5 argument(s)
─( 15:43:58 )─< command 1 >──────────────────────────────────────{ counter: 0 }─
─( 15:43:58 )─< 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
File "binary_tree.ml", line 22, characters 9-41:
Error: The constructor Fork expects 3 argument(s),
       but is applied here to 5 argument(s)
─( 15:43:58 )─< command 2 >──────────────────────────────────────{ 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 1, Fork (6, Leaf 5, Leaf 5))
─( 15:44:07 )─< command 3 >──────────────────────────────────────{ 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 1, Fork (6, Leaf 5, Leaf 5))
val t3 : string tree = Fork ("Hello", Leaf "World", Leaf "!")
─( 15:45:01 )─< command 4 >──────────────────────────────────────{ counter: 0 }─
utop # [1;2;3] ;;
- : int list = [1; 2; 3]
─( 15:45:44 )─< command 5 >──────────────────────────────────────{ counter: 0 }─
utop # ["Hello"; "World"];;
- : string list = ["Hello"; "World"]
─( 15:45:53 )─< command 6 >──────────────────────────────────────{ counter: 0 }─
utop # #use "binary_tree.ml";;
type 'a tree = Leaf of 'a | Fork of 'a * 'a tree * 'a tree
type inttree = ILeaf of int | IFork of int * inttree * inttree
val t1 : int tree = Leaf 5
val t2 : int tree = Fork (3, Leaf 1, Fork (6, Leaf 5, Leaf 5))
val t3 : string tree = Fork ("Hello", Leaf "World", Leaf "!")
─( 15:46:00 )─< command 7 >──────────────────────────────────────{ 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 1, Fork (6, Leaf 5, Leaf 5))
val t3 : string tree = Fork ("Hello", Leaf "World", Leaf "!")
val tsize : 'a tree -> int = <fun>
val tsum : int tree -> int = <fun>
val tsum' : int tree -> int = <fun>
type inttree = ILeaf of int | IFork of int * inttree * inttree
-( 15:47:53 )-< command 8 >-------------------------------------{ counter: 0 }-
utop # tsize t3 ;;
- : int = 3
-( 15:55:48 )-< command 9 >-------------------------------------{ counter: 0 }-
utop # t3 ;;
- : string tree = Fork ("Hello", Leaf "World", Leaf "!")
-( 15:55:54 )-< command 10 >------------------------------------{ counter: 0 }-
utop # t2 ;;
- : int tree = Fork (3, Leaf 1, Fork (6, Leaf 5, Leaf 5))
-( 15:55:56 )-< command 11 >------------------------------------{ counter: 0 }-
utop # tsize t2 ;;
- : int = 5
-( 15:55:58 )-< command 12 >------------------------------------{ counter: 0 }-
utop # tsum t2 ;;
- : int = 20
-( 15:56:01 )-< command 13 >------------------------------------{ counter: 0 }-
utop # tsize ;;
- : 'a tree -> int = <fun>
-( 15:56:05 )-< command 14 >------------------------------------{ counter: 0 }-
utop # tsum ;;
- : int tree -> int = <fun>
-( 15:56:10 )-< command 15 >------------------------------------{ 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 1, Fork (6, Leaf 5, Leaf 5))
val t3 : string tree = Fork ("Hello", Leaf "World", Leaf "!")
File "binary_tree.ml", line 29, characters 41-43:
Error: This expression has type 'a tree but an expression was expected of type
         'a -> 'b
-( 15:56:12 )-< command 16 >------------------------------------{ 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 1, Fork (6, Leaf 5, Leaf 5))
val t3 : string tree = Fork ("Hello", Leaf "World", Leaf "!")
val tmap : ('a -> 'b) -> 'a tree -> 'b tree = <fun>
val tsize : 'a tree -> int = <fun>
val tsum : int tree -> int = <fun>
val tsum' : int tree -> int = <fun>
type inttree = ILeaf of int | IFork of int * inttree * inttree
-( 16:00:58 )-< command 17 >------------------------------------{ counter: 0 }-
utop # t2 ;;
- : int tree = Fork (3, Leaf 1, Fork (6, Leaf 5, Leaf 5))
-( 16:01:11 )-< command 18 >------------------------------------{ counter: 0 }-
utop # tmap (fun x -> x + 1)  t2 ;;
```

```
- : int tree = Fork (4, Leaf 2, Fork (7, Leaf 6, Leaf 6))
-( 16:01:14 )-< command 19 >-------------------------------------{ counter: 0 }-
utop # tmap String.length t3 ;;
- : int tree = Fork (5, Leaf 5, Leaf 1)
-( 16:01:40 )-< command 20 >-------------------------------------{ counter: 0 }-
utop # t3 ;;
- : string tree = Fork ("Hello", Leaf "World", Leaf "!")
-( 16:01:49 )-< command 21 >-------------------------------------{ counter: 0 }-
utop # tmap ;;
- : ('a -> 'b) -> 'a tree -> 'b tree = <fun>
-( 16:01:51 )-< command 22 >-------------------------------------{ counter: 0 }-
utop # List.fold_right ;;
- : ('a -> 'b -> 'b) -> 'a list -> 'b -> 'b = <fun>
-( 16:02:05 )-< command 23 >-------------------------------------{ counter: 0 }-
utop # List.fold_right (+) (1::2::3::4::[]) 0 ;;
- : int = 10
-( 16:09:16 )-< command 24 >-------------------------------------{ counter: 0 }-
utop # List.fold_right (fun h t -> h :: t) (1::2::3::4::[]) [] ;;
- : int list = [1; 2; 3; 4]
-( 16:11:06 )-< command 25 >-------------------------------------{ 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 1, Fork (6, Leaf 5, Leaf 5))
val t3 : string tree = Fork ("Hello", Leaf "World", Leaf "!")
val tmap : ('a -> 'b) -> 'a tree -> 'b tree = <fun>
File "binary_tree.ml", line 33, characters 19-30:
Error: This expression has type 'a tree but an expression was expected of type
         'a
       The type variable 'a occurs inside 'a tree
-( 16:12:30 )-< command 26 >-------------------------------------{ 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 1, Fork (6, 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 tsize : 'a tree -> int = <fun>
val tsum : int tree -> int = <fun>
val tsum' : int tree -> int = <fun>
type inttree = ILeaf of int | IFork of int * inttree * inttree
-( 16:14:15 )-< command 27 >-------------------------------------{ counter: 0 }-
utop # tfold (+) 0 t2 ;;
- : int = 20
-( 16:14:42 )-< command 28 >-------------------------------------{ counter: 0 }-
utop # t2 ;;
- : int tree = Fork (3, Leaf 1, Fork (6, Leaf 5, Leaf 5))
-( 16:14:55 )-< command 29 >-------------------------------------{ counter: 0 }-
utop # List.fold_right (fun h t -> h :: t) (1::2::3::4::[]) [] ;;
- : int list = [1; 2; 3; 4]
-( 16:14:57 )-< command 30 >-------------------------------------{ counter: 0 }-
utop # List.fold_right ;;
```

```
- : ('a -> 'b -> 'b) -> 'a list -> 'b -> 'b = <fun>
-( 16:16:51 )-< command 31 >-------------------------------------{ counter: 0 }-
utop # #use "binary_tree.ml";;
type 'a tree = Leaf of 'a | Fork of 'a * 'a tree * 'a tree
val tfold : ('a -> 'b) -> ('a -> 'b -> 'b -> 'b) -> 'a tree -> 'b = <fun>
val t1 : int tree = Leaf 5
val t2 : int tree = Fork (3, Leaf 1, Fork (6, Leaf 5, Leaf 5))
val t3 : string tree = Fork ("Hello", Leaf "World", Leaf "!")
val tmap : ('a -> 'b) -> 'a tree -> 'b tree = <fun>
val tsize : 'a tree -> int = <fun>
val tsum : int tree -> int = <fun>
val tsum' : int tree -> int = <fun>
type inttree = ILeaf of int | IFork of int * inttree * inttree
-( 16:17:41 )-< command 32 >-------------------------------------{ counter: 0 }-
utop # t2 ;;
- : int tree = Fork (3, Leaf 1, Fork (6, Leaf 5, Leaf 5))
-( 16:20:47 )-< command 33 >-------------------------------------{ counter: 0 }-
utop # tfold ;;
- : ('a -> 'b) -> ('a -> 'b -> 'b -> 'b) -> 'a tree -> 'b = <fun>
-( 16:21:07 )-< command 34 >-------------------------------------{ counter: 0 }-
utop # tfold (fun x -> x) (fun x y z -> x + y + z) t2 ;;
- : int = 20
-( 16:21:53 )-< command 35 >-------------------------------------{ counter: 0 }-
utop # t3 ;;
- : string tree = Fork ("Hello", Leaf "World", Leaf "!")
-( 16:23:23 )-< command 36 >-------------------------------------{ counter: 0 }-
utop #
```

Arg	Arith_status	Array	ArrayLabels	Assert_failure	Big_int	Bigarray	Buffer	Call