```
Last login: Mon Apr 10 15:35:53 on ttys005
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 "wolf.ml";;
val is_not_elem : 'a list -> 'a -> bool = <fun>
val run : 'a -> unit = <fun>
val is_elem : 'a -> 'a list -> bool = <fun>
type loc = L | R
type state = loc * loc * loc * loc
val ok_state : state -> bool = <fun>
val final : loc * loc * loc * loc -> bool = <fun>
val other_side : loc -> loc = <fun>
val moves : state -> state list = <fun>
─( 16:10:35 )─< command 1 >──────────────────────────────────────{ counter: 0 }─
utop # moves (L,L,L,L) ;;
- : state list = [(R, L, R, L)]
─( 16:10:39 )─< command 2 >──────────────────────────────────────{ counter: 0 }─
utop # List.map ok_state (moves (L,L,L,L)) ;;
- : bool list = [true]
─( 16:10:47 )─< command 3 >──────────────────────────────────────{ counter: 0 }─
utop # List.map final (moves (L,L,L,L)) ;;
- : bool list = [false]
─( 16:11:30 )─< command 4 >──────────────────────────────────────{ counter: 0 }─
utop # #use "wolf.ml";;
val is_not_elem : 'a list -> 'a -> bool = <fun>
val run : 'a -> unit = <fun>
val is_elem : 'a -> 'a list -> bool = <fun>
type loc = L | R
type state = loc * loc * loc * loc
val ok_state : state -> bool = <fun>
val final : loc * loc * loc * loc -> bool = <fun>
val other_side : loc -> loc = <fun>
val moves : state -> state list = <fun>
val crossing_v1 : unit -> state list option = <fun>
─( 16:11:42 )─< command 5 >──────────────────────────────────────{ counter: 0 }─
utop # crossing_v1 () ;;
- : state list option =
Some
 [(L, L, L, L); (R, L, R, L); (L, L, R, L); (R, R, R, L); (L, R, L, L);
  (R, R, L, R); (L, R, L, R); (R, R, R, R)]
─( 16:18:12 )─< command 6 >──────────────────────────────────────{ counter: 0 }─
utop # #use "wolf.ml";;
val is_not_elem : 'a list -> 'a -> bool = <fun>
val run : 'a -> unit = <fun>
val is_elem : 'a -> 'a list -> bool = <fun>
type loc = L | R
type state = loc * loc * loc * loc
val ok_state : state -> bool = <fun>
val final : loc * loc * loc * loc -> bool = <fun>
val other_side : loc -> loc = <fun>
val moves : state -> state list = <fun>
val crossing_v1 : unit -> state list option = <fun>
exception FoundPath of (loc * loc * loc * loc) list
val crossing_v2 : unit -> (loc * loc * loc * loc) list option = <fun>
```

```
─( 16:18:42 )─< command 7 >──────────────────────────────────────{ counter: 0 }─
utop # crossing_v2 () ;;
- : (loc * loc * loc * loc) list option =
Some
 [(L, L, L, L); (R, L, R, L); (L, L, R, L); (R, R, R, L); (L, R, L, L);
  (R, R, L, R); (L, R, L, R); (R, R, R, R)]
─( 16:20:46 )─< command 8 >──────────────────────────────────────{ counter: 0 }─
utop # #use "wolf.ml";;
val explode : string -> char list = <fun>
val is_not_elem : 'a list -> 'a -> bool = <fun>
val run : 'a -> unit = <fun>
val is_elem : 'a -> 'a list -> bool = <fun>
type loc = L | R
type state = loc * loc * loc * loc
val ok_state : state -> bool = <fun>
val final : loc * loc * loc * loc -> bool = <fun>
val other_side : loc -> loc = <fun>
val moves : state -> state list = <fun>
val crossing_v1 : unit -> state list option = <fun>
exception FoundPath of (loc * loc * loc * loc) list
val crossing_v2 : unit -> (loc * loc * loc * loc) list option = <fun>
exception KeepLooking
val process_solution_exn : ('a -> string) -> 'a -> 'a option = <fun>
val show_list : ('a -> string) -> 'a list -> string = <fun>
val show_loc : loc -> string = <fun>
val show_state : loc * loc * loc * loc -> string = <fun>
val show_path : (loc * loc * loc * loc) list -> string = <fun>
val crossing_v3 : unit -> state list option = <fun>
─( 16:20:52 )─< command 9 >──────────────────────────────────────{ counter: 0 }─
utop # crossing_v3 () ;;
Here is a solution:
[ (L, L, L, L); (R, L, R, L); (L, L, R, L); (R, R, R, L); (L, R, L, L); (R, R, L, R); (L,
R, L, R); (R, R, R, R) ]
Do you like it?
n
Here is a solution:
[ (L, L, L, L); (R, L, R, L); (L, L, R, L); (R, L, R, R); (L, L, L, R); (R, R, L, R); (L,
R, L, R); (R, R, R, R) ]
Do you like it?
n
- : state list option = None
─( 16:23:18 )─< command 10 >─────────────────────────────────────{ counter: 0 }─
utop #
```

Arg	Arith_status	Array	ArrayLabels	Assert_failure	Big_int	Bigarray	Buffer	Callback	Camli