```
Last login: Mon Apr 10 13:19:27 on ttys005
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 "subsetsum_cps.ml";;
val show_list : ('a -> string) -> 'a list -> string = <fun>
val is_elem : 'a -> 'a list -> bool = <fun>
val sum : int list -> int = <fun>
File "subsetsum_cps.ml", line 47, characters 21-28:
Error: Unbound value explode
─( 13:34:48 )─< command 1 >──────────────────────────────────────────{ counter: 0 }─
utop # #use "subsetsum_cps.ml";;
val explode : string -> char list = <fun>
val show_list : ('a -> string) -> 'a list -> string = <fun>
val is_elem : 'a -> 'a list -> bool = <fun>
val sum : int list -> int = <fun>
val process_solution_cps_v1 : ('a -> string) -> 'a -> (unit -> 'b) -> (unit -> 'b) -> 'b =
  <fun>
val try_subset_cps_v1 : int list -> int list -> (unit -> 'a) -> (unit -> 'a) -> 'a = <fun>
val subsetsum_cps_v1 : int list -> unit = <fun>
─( 13:34:50 )─< command 2 >──────────────────────────────────────────{ counter: 0 }─
utop # subsetsum_cps_v1 [ 1; -2; 5; -5 ; 6] ;;
Here is a solution:
[ 1; -2; -5; 6 ]
Do you like it?
y
Yeah, we found one
- : unit = ()
─( 13:35:27 )─< command 3 >──────────────────────────────────────────{ counter: 0 }─
utop # subsetsum_cps_v1 [ 1; -2; 5; -5 ; 6] ;;
Here is a solution:
[ 1; -2; -5; 6 ]
Do you like it?
n
Here is a solution:
[ 5; -5 ]
Do you like it?
n
Oh no, no subset found.
- : unit = ()
─( 13:35:53 )─< command 4 >──────────────────────────────────────────{ counter: 0 }─
utop # #use "subsetsum_cps.ml";;
val explode : string -> char list = <fun>
val show_list : ('a -> string) -> 'a list -> string = <fun>
val is_elem : 'a -> 'a list -> bool = <fun>
val sum : int list -> int = <fun>
val process_solution_cps_v1 : ('a -> string) -> 'a -> (unit -> 'b) -> (unit -> 'b) -> 'b =
  <fun>
val try_subset_cps_v1 : int list -> int list -> (unit -> 'a) -> (unit -> 'a) -> 'a = <fun>
val subsetsum_cps_v1 : int list -> unit = <fun>
val process_solution_cps_v2 : ('a -> string) -> 'a -> ('a -> 'b) -> (unit -> 'b) -> 'b =
  <fun>
val try_subset_cps_v2 : int list -> int list -> (int list -> 'a) -> (unit -> 'a) -> 'a =
  <fun>
val subsetsum_cps_v2 : int list -> unit = <fun>
─( 13:36:07 )─< command 5 >──────────────────────────────────────────{ counter: 0 }─
utop # subsetsum_cps_v2 [ 1; -2; 3; -5; 6 ] ;;
Here is a solution:
[ 1; -2; -5; 6 ]
Do you like it?
y
Yeah, we found one.
It is as folows:
[ 1; -2; -5; 6 ]
- : unit = ()
─( 13:37:59 )─< 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>
─( 13:38:13 )─< command 7 >──────────────────────────────────────────{ counter: 0 }─
utop # moves (L,L,L,L) ;;
- : state list = [(R, L, R, L)]
─( 13:56:34 )─< command 8 >──────────────────────────────────────────{ counter: 0 }─
utop # List.map final (moves (L,L,L,L)) ;;
- : bool list = [false]
─( 13:56:44 )─< command 9 >──────────────────────────────────────────{ counter: 0 }─
utop # moves (L,L,L,L) ;;
- : state list = [(R, L, R, L)]
─( 13:57:38 )─< command 10 >─────────────────────────────────────────{ counter: 0 }─
utop # moves (List.hd (moves (L,L,L,L))) ;;
- : state list = [(L, L, R, L); (L, L, L, L)]
─( 13:58:11 )─< command 11 >─────────────────────────────────────────{ 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>
─( 13:58:29 )─< command 12 >─────────────────────────────────────────{ 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)]
─( 14:04:52 )─< command 13 >─────────────────────────────────────────{ 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>
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>
─( 14:05:11 )─< command 14 >─────────────────────────────────────────────────{ 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
─( 14:11:41 )─< command 15 >─────────────────────────────────────────────────{ counter: 0 }─
utop #
```

Arg	Arith_status	Array	ArrayLabels	Assert_failure	Big_int	Bigarray	Buffer	Callback	Camlint