Containers
CCArrayLabels.(--) : int -> int -> int t
CCArrayLabels.(--^) : int -> int -> int t
CCArrayLabels.(>>=) : 'a t -> ('a -> 'b t) -> 'b t
CCArrayLabels.(>>|) : 'a t -> ('a -> 'b) -> 'b t
CCArrayLabels.(>|=) : 'a t -> ('a -> 'b) -> 'b t
CCArrayLabels.(and*) : 'a array -> 'b array -> ('a * 'b) array
CCArrayLabels.(and+) : 'a array -> 'b array -> ('a * 'b) array
CCArrayLabels.(let*) : 'a array -> ('a -> 'b array) -> 'b array
CCArrayLabels.(let+) : 'a array -> ('a -> 'b) -> 'b array
CCArrayLabels.append : 'a array -> 'a array -> 'a array
CCArrayLabels.blit : src:'a array -> src_pos:int -> dst:'a array -> dst_pos:int -> len:int -> unit
CCArrayLabels.bsearch : cmp:('a -> 'a -> int) -> key:'a -> 'a t -> [`All_bigger | `All_lower | `At of int | `Empty | `Just_after of int]
CCArrayLabels.combine : 'a array -> 'b array -> ('a * 'b) array
CCArrayLabels.compare : 'a ord -> 'a t ord
CCArrayLabels.concat : 'a array list -> 'a array
CCArrayLabels.copy : 'a array -> 'a array
CCArrayLabels.create_matrix : dimx:int -> dimy:int -> 'a -> 'a array array
CCArrayLabels.empty : 'a t
CCArrayLabels.equal : 'a equal -> 'a t equal
CCArrayLabels.except_idx : 'a t -> int -> 'a list
CCArrayLabels.exists : f:('a -> bool) -> 'a array -> bool
CCArrayLabels.exists2 : f:('a -> 'b -> bool) -> 'a t -> 'b t -> bool
CCArrayLabels.fast_sort : cmp:('a -> 'a -> int) -> 'a array -> unit
CCArrayLabels.fill : 'a array -> pos:int -> len:int -> 'a -> unit
CCArrayLabels.filter : f:('a -> bool) -> 'a t -> 'a t
CCArrayLabels.filter_map : f:('a -> 'b option) -> 'a t -> 'b t
CCArrayLabels.find_idx : f:('a -> bool) -> 'a t -> (int * 'a) option
CCArrayLabels.find_map : f:('a -> 'b option) -> 'a t -> 'b option
CCArrayLabels.find_map_i : f:(int -> 'a -> 'b option) -> 'a t -> 'b option
CCArrayLabels.find_opt : f:('a -> bool) -> 'a array -> 'a option
CCArrayLabels.flat_map : f:('a -> 'b t) -> 'a t -> 'b array
CCArrayLabels.fold : f:('a -> 'b -> 'a) -> init:'a -> 'b t -> 'a
CCArrayLabels.fold2 : f:('acc -> 'a -> 'b -> 'acc) -> init:'acc -> 'a t -> 'b t -> 'acc
CCArrayLabels.fold_left : f:('a -> 'b -> 'a) -> init:'a -> 'b array -> 'a

Containers
CCArrayLabels.fold_left_map : f:('a -> 'b -> 'a * 'c) -> init:'a -> 'b array -> 'a * 'c array
CCArrayLabels.fold_map : f:('acc -> 'a -> 'acc * 'b) -> init:'acc -> 'a t -> 'acc * 'b t
CCArrayLabels.fold_right : f:('b -> 'a -> 'a) -> 'b array -> init:'a -> 'a
CCArrayLabels.fold_while : f:('a -> 'b -> 'a * [`Continue | `Stop]) -> init:'a -> 'b t -> 'a
CCArrayLabels.foldi : f:('a -> int -> 'b -> 'a) -> init:'a -> 'b t -> 'a
CCArrayLabels.for_all : f:('a -> bool) -> 'a array -> bool
CCArrayLabels.for_all2 : f:('a -> 'b -> bool) -> 'a t -> 'b t -> bool
CCArrayLabels.get_safe : 'a t -> int -> 'a option
CCArrayLabels.init : int -> f:(int -> 'a) -> 'a array
CCArrayLabels.iter : f:('a -> unit) -> 'a array -> unit
CCArrayLabels.iter2 : f:('a -> 'b -> unit) -> 'a t -> 'b t -> unit
CCArrayLabels.iteri : f:(int -> 'a -> unit) -> 'a array -> unit
CCArrayLabels.lookup : cmp:'a ord -> key:'a -> 'a t -> int option
CCArrayLabels.lookup_exn : cmp:'a ord -> key:'a -> 'a t -> int
CCArrayLabels.make_float : int -> float array
CCArrayLabels.make_matrix : dimx:int -> dimy:int -> 'a -> 'a array array
CCArrayLabels.map : f:('a -> 'b) -> 'a array -> 'b array
CCArrayLabels.map2 : f:('a -> 'b -> 'c) -> 'a t -> 'b t -> 'c t
CCArrayLabels.map_inplace : f:('a -> 'a) -> 'a t -> unit
CCArrayLabels.mapi : f:(int -> 'a -> 'b) -> 'a array -> 'b array
CCArrayLabels.mem : ?eq:('a -> 'a -> bool) -> 'a -> 'a t -> bool
CCArrayLabels.memq : 'a -> set:'a array -> bool
CCArrayLabels.monoid_product : f:('a -> 'b -> 'c) -> 'a t -> 'b t -> 'c t
CCArrayLabels.of_list : 'a list -> 'a array
CCArrayLabels.of_seq : 'a Seq.t -> 'a array
CCArrayLabels.pp : ?pp_start:unit printer -> ?pp_stop:unit printer -> ?pp_sep:unit printer -> 'a printer -> 'a t printer
CCArrayLabels.pp_i : ?pp_start:unit printer -> ?pp_stop:unit printer -> ?pp_sep:unit printer -> (int -> 'a printer) -> 'a t printer
CCArrayLabels.random : 'a random_gen -> 'a t random_gen
CCArrayLabels.random_choose : 'a t -> 'a random_gen
CCArrayLabels.random_len : int -> 'a random_gen -> 'a t random_gen
CCArrayLabels.random_non_empty : 'a random_gen -> 'a t random_gen
CCArrayLabels.rev : 'a t -> 'a t
CCArrayLabels.reverse_in_place : 'a t -> unit
CCArrayLabels.scan_left : f:('acc -> 'a -> 'acc) -> init:'acc -> 'a t -> 'acc t

Containers
CCArrayLabels.shuffle : 'a t -> unit
CCArrayLabels.shuffle_with : Random.State.t -> 'a t -> unit
CCArrayLabels.sort : cmp:('a -> 'a -> int) -> 'a array -> unit
CCArrayLabels.sort_generic : (module MONO_ARRAY with type elt = 'elt and type t = 'arr) -> cmp:('elt -> 'elt -> int) -> 'arr -> unit
CCArrayLabels.sort_indices : f:('a -> 'a -> int) -> 'a t -> int array
CCArrayLabels.sort_ranking : f:('a -> 'a -> int) -> 'a t -> int array
CCArrayLabels.sorted : f:('a -> 'a -> int) -> 'a t -> 'a array
CCArrayLabels.split : ('a * 'b) array -> 'a array * 'b array
CCArrayLabels.stable_sort : cmp:('a -> 'a -> int) -> 'a array -> unit
CCArrayLabels.sub : 'a array -> pos:int -> len:int -> 'a array
CCArrayLabels.swap : 'a t -> int -> int -> unit
CCArrayLabels.to_gen : 'a t -> 'a gen
CCArrayLabels.to_iter : 'a t -> 'a iter
CCArrayLabels.to_list : 'a array -> 'a list
CCArrayLabels.to_seq : 'a t -> 'a Seq.t
CCArrayLabels.to_seqi : 'a array -> (int * 'a) Seq.t
CCArrayLabels.to_string : ?sep:string -> ('a -> string) -> 'a array -> string
CCArray.(--) : int -> int -> int t
CCArray.(--^) : int -> int -> int t
CCArray.(>>=) : 'a t -> ('a -> 'b t) -> 'b t
CCArray.(>>|) : 'a t -> ('a -> 'b) -> 'b t
CCArray.(>|=) : 'a t -> ('a -> 'b) -> 'b t
CCArray.(and*) : 'a array -> 'b array -> ('a * 'b) array
CCArray.(and+) : 'a array -> 'b array -> ('a * 'b) array
CCArray.(let*) : 'a array -> ('a -> 'b array) -> 'b array
CCArray.(let+) : 'a array -> ('a -> 'b) -> 'b array
CCArray.append : 'a array -> 'a array -> 'a array
CCArray.blit : 'a array -> int -> 'a array -> int -> int -> unit
CCArray.bsearch : cmp:('a -> 'a -> int) -> 'a -> 'a t -> [`All_bigger | `All_lower | `At of int | `Empty | `Just_after of int]
CCArray.combine : 'a array -> 'b array -> ('a * 'b) array
CCArray.compare : 'a ord -> 'a t ord
CCArray.concat : 'a array list -> 'a array
CCArray.copy : 'a array -> 'a array
CCArray.create_matrix : int -> int -> 'a -> 'a array array

Containers
CCArray.empty : 'a t
CCArray.equal : 'a equal -> 'a t equal
CCArray.except_idx : 'a t -> int -> 'a list
CCArray.exists : ('a -> bool) -> 'a array -> bool
CCArray.exists2 : ('a -> 'b -> bool) -> 'a t -> 'b t -> bool
CCArray.fast_sort : ('a -> 'a -> int) -> 'a array -> unit
CCArray.fill : 'a array -> int -> int -> 'a -> unit
CCArray.filter : ('a -> bool) -> 'a t -> 'a t
CCArray.filter_map : ('a -> 'b option) -> 'a t -> 'b t
CCArray.find_idx : ('a -> bool) -> 'a t -> (int * 'a) option
CCArray.find_map : ('a -> 'b option) -> 'a t -> 'b option
CCArray.find_map_i : (int -> 'a -> 'b option) -> 'a t -> 'b option
CCArray.find_opt : ('a -> bool) -> 'a array -> 'a option
CCArray.flat_map : ('a -> 'b t) -> 'a t -> 'b array
CCArray.fold : ('a -> 'b -> 'a) -> 'a -> 'b t -> 'a
CCArray.fold2 : ('acc -> 'a -> 'b -> 'acc) -> 'acc -> 'a t -> 'b t -> 'acc
CCArray.fold_left : ('a -> 'b -> 'a) -> 'a -> 'b array -> 'a
CCArray.fold_left_map : ('a -> 'b -> 'a * 'c) -> 'a -> 'b array -> 'a * 'c array
CCArray.fold_map : ('acc -> 'a -> 'acc * 'b) -> 'acc -> 'a t -> 'acc * 'b t
CCArray.fold_right : ('b -> 'a -> 'a) -> 'b array -> 'a -> 'a
CCArray.fold_while : ('a -> 'b -> 'a * [`Continue | `Stop]) -> 'a -> 'b t -> 'a
CCArray.foldi : ('a -> int -> 'b -> 'a) -> 'a -> 'b t -> 'a
CCArray.for_all : ('a -> bool) -> 'a array -> bool
CCArray.for_all2 : ('a -> 'b -> bool) -> 'a t -> 'b t -> bool
CCArray.get_safe : 'a t -> int -> 'a option
CCArray.init : int -> (int -> 'a) -> 'a array
CCArray.iter : ('a -> unit) -> 'a array -> unit
CCArray.iter2 : ('a -> 'b -> unit) -> 'a array -> 'b array -> unit
CCArray.iteri : (int -> 'a -> unit) -> 'a array -> unit
CCArray.lookup : cmp:'a ord -> 'a -> 'a t -> int option
CCArray.lookup_exn : cmp:'a ord -> 'a -> 'a t -> int
CCArray.make_float : int -> float array
CCArray.make_matrix : int -> int -> 'a -> 'a array array
CCArray.map : ('a -> 'b) -> 'a array -> 'b array

Containers
CCArray.map2 : ('a -> 'b -> 'c) -> 'a array -> 'b array -> 'c array
CCArray.map_inplace : ('a -> 'a) -> 'a array -> unit
CCArray.mapi : (int -> 'a -> 'b) -> 'a array -> 'b array
CCArray.mem : ?eq:('a -> 'a -> bool) -> 'a -> 'a t -> bool
CCArray.memq : 'a -> 'a array -> bool
CCArray.monoid_product : ('a -> 'b -> 'c) -> 'a t -> 'b t -> 'c t
CCArray.of_list : 'a list -> 'a array
CCArray.of_seq : 'a Seq.t -> 'a array
CCArray.pp : ?pp_start:unit printer -> ?pp_stop:unit printer -> ?pp_sep:unit printer -> 'a printer -> 'a t printer
CCArray.pp_i : ?pp_start:unit printer -> ?pp_stop:unit printer -> ?pp_sep:unit printer -> (int -> 'a printer) -> 'a t printer
CCArray.random : 'a random_gen -> 'a t random_gen
CCArray.random_choose : 'a t -> 'a random_gen
CCArray.random_len : int -> 'a random_gen -> 'a t random_gen
CCArray.random_non_empty : 'a random_gen -> 'a t random_gen
CCArray.rev : 'a t -> 'a t
CCArray.reverse_in_place : 'a t -> unit
CCArray.scan_left : ('acc -> 'a -> 'acc) -> 'acc -> 'a t -> 'acc t
CCArray.shuffle : 'a t -> unit
CCArray.shuffle_with : Random.State.t -> 'a t -> unit
CCArray.sort : ('a -> 'a -> int) -> 'a array -> unit
CCArray.sort_generic : (module MONO_ARRAY with type elt = 'elt and type t = 'arr) -> cmp:('elt -> 'elt -> int) -> 'arr -> unit
CCArray.sort_indices : ('a -> 'a -> int) -> 'a t -> int array
CCArray.sort_ranking : ('a -> 'a -> int) -> 'a t -> int array
CCArray.sorted : ('a -> 'a -> int) -> 'a t -> 'a array
CCArray.split : ('a * 'b) array -> 'a array * 'b array
CCArray.stable_sort : ('a -> 'a -> int) -> 'a array -> unit
CCArray.sub : 'a array -> int -> int -> 'a array
CCArray.swap : 'a t -> int -> int -> unit
CCArray.to_gen : 'a t -> 'a gen
CCArray.to_iter : 'a t -> 'a iter
CCArray.to_list : 'a array -> 'a list
CCArray.to_seq : 'a t -> 'a Seq.t
CCArray.to_seqi : 'a array -> (int * 'a) Seq.t
CCArray.to_string : ?sep:string -> ('a -> string) -> 'a array -> string

Containers
CCListLabels.(--) : int -> int -> int CCList.t
CCListLabels.(--^) : int -> int -> int CCList.t
CCListLabels.(<$>) : ('a -> 'b) -> 'a CCList.t -> 'b CCList.t
CCListLabels.(<*>) : ('a -> 'b) CCList.t -> 'a CCList.t -> 'b CCList.t
CCListLabels.(>>=) : 'a CCList.t -> ('a -> 'b CCList.t) -> 'b CCList.t
CCListLabels.(>|=) : 'a CCList.t -> ('a -> 'b) -> 'b CCList.t
CCListLabels.(@) : 'a CCList.t -> 'a CCList.t -> 'a CCList.t
CCListLabels.(and&) : 'a list -> 'b list -> ('a * 'b) list
CCListLabels.(and*) : 'a CCList.t -> 'b CCList.t -> ('a * 'b) CCList.t
CCListLabels.(and+) : 'a CCList.t -> 'b CCList.t -> ('a * 'b) CCList.t
CCListLabels.(let*) : 'a CCList.t -> ('a -> 'b CCList.t) -> 'b CCList.t
CCListLabels.(let+) : 'a CCList.t -> ('a -> 'b) -> 'b CCList.t
CCListLabels.Assoc.get : eq:('a -> 'a -> bool) -> 'a -> ('a, 'b) t -> 'b option
CCListLabels.Assoc.get_exn : eq:('a -> 'a -> bool) -> 'a -> ('a, 'b) t -> 'b
CCListLabels.Assoc.keys : ('a, 'b) t -> 'a list
CCListLabels.Assoc.map_values : ('b -> 'c) -> ('a, 'b) t -> ('a, 'c) t
CCListLabels.Assoc.set : eq:('a -> 'a -> bool) -> 'a -> 'b -> ('a, 'b) t -> ('a, 'b) t
CCListLabels.Assoc.update : eq:('a -> 'a -> bool) -> f:('b option -> 'b option) -> 'a -> ('a, 'b) t -> ('a, 'b) t
CCListLabels.Assoc.values : ('a, 'b) t -> 'b list
CCListLabels.Ref.clear : 'a t -> unit
CCListLabels.Ref.create : unit -> 'a t
CCListLabels.Ref.lift : ('a list -> 'b) -> 'a t -> 'b
CCListLabels.Ref.pop : 'a t -> 'a option
CCListLabels.Ref.pop_exn : 'a t -> 'a
CCListLabels.Ref.push : 'a t -> 'a -> unit
CCListLabels.Ref.push_list : 'a t -> 'a list -> unit
CCListLabels.add_nodup : eq:('a -> 'a -> bool) -> 'a -> 'a t -> 'a t
CCListLabels.all_ok : ('a, 'err) result t -> ('a t, 'err) result
CCListLabels.all_some : 'a option t -> 'a t option
CCListLabels.append : 'a t -> 'a t -> 'a t
CCListLabels.assoc : eq:('a -> 'a -> bool) -> 'a -> ('a * 'b) t -> 'b
CCListLabels.assoc_opt : eq:('a -> 'a -> bool) -> 'a -> ('a * 'b) t -> 'b option
CCListLabels.assq : 'a -> ('a * 'b) list -> 'b
CCListLabels.assq_opt : 'a -> ('a * 'b) t -> 'b option

Containers
CCListLabels.cartesian_product : 'a t t -> 'a t t
CCListLabels.chunks : int -> 'a list -> 'a list list
CCListLabels.combine : 'a list -> 'b list -> ('a * 'b) list
CCListLabels.combine_gen : 'a list -> 'b list -> ('a * 'b) gen
CCListLabels.combine_shortest : 'a list -> 'b list -> ('a * 'b) list
CCListLabels.compare : ('a -> 'a -> int) -> 'a t -> 'a t -> int
CCListLabels.compare_length_with : 'a t -> int -> int
CCListLabels.compare_lengths : 'a t -> 'b t -> int
CCListLabels.concat : 'a list list -> 'a list
CCListLabels.concat_map : f:('a -> 'b list) -> 'a list -> 'b list
CCListLabels.cons : 'a -> 'a t -> 'a t
CCListLabels.cons' : 'a t -> 'a -> 'a t
CCListLabels.cons_maybe : 'a option -> 'a t -> 'a t
CCListLabels.count : f:('a -> bool) -> 'a list -> int
CCListLabels.count_true_false : f:('a -> bool) -> 'a list -> int * int
CCListLabels.diagonal : 'a t -> ('a * 'a) t
CCListLabels.drop : int -> 'a t -> 'a t
CCListLabels.drop_while : f:('a -> bool) -> 'a t -> 'a t
CCListLabels.empty : 'a t
CCListLabels.equal : ('a -> 'a -> bool) -> 'a t -> 'a t -> bool
CCListLabels.exists : f:('a -> bool) -> 'a list -> bool
CCListLabels.exists2 : f:('a -> 'b -> bool) -> 'a list -> 'b list -> bool
CCListLabels.fast_sort : cmp:('a -> 'a -> int) -> 'a list -> 'a list
CCListLabels.filter : f:('a -> bool) -> 'a t -> 'a t
CCListLabels.filter_map : f:('a -> 'b option) -> 'a t -> 'b t
CCListLabels.filteri : f:(int -> 'a -> bool) -> 'a list -> 'a list
CCListLabels.find : f:('a -> bool) -> 'a list -> 'a
CCListLabels.find_all : f:('a -> bool) -> 'a list -> 'a list
CCListLabels.find_idx : f:('a -> bool) -> 'a t -> (int * 'a) option
CCListLabels.find_map : f:('a -> 'b option) -> 'a t -> 'b option
CCListLabels.find_mapi : f:(int -> 'a -> 'b option) -> 'a t -> 'b option
CCListLabels.find_opt : f:('a -> bool) -> 'a t -> 'a option
CCListLabels.find_pred : f:('a -> bool) -> 'a t -> 'a option
CCListLabels.find_pred_exn : f:('a -> bool) -> 'a t -> 'a

Containers
CCListLabels.flat_map : f:('a -> 'b t) -> 'a t -> 'b t
CCListLabels.flat_map_i : f:(int -> 'a -> 'b t) -> 'a t -> 'b t
CCListLabels.flatten : 'a t t -> 'a t
CCListLabels.fold_filter_map : f:('acc -> 'a -> 'acc * 'b option) -> init:'acc -> 'a list -> 'acc * 'b list
CCListLabels.fold_filter_map_i : f:('acc -> int -> 'a -> 'acc * 'b option) -> init:'acc -> 'a list -> 'acc * 'b list
CCListLabels.fold_flat_map : f:('acc -> 'a -> 'acc * 'b list) -> init:'acc -> 'a list -> 'acc * 'b list
CCListLabels.fold_flat_map_i : f:('acc -> int -> 'a -> 'acc * 'b list) -> init:'acc -> 'a list -> 'acc * 'b list
CCListLabels.fold_left : f:('a -> 'b -> 'a) -> init:'a -> 'b list -> 'a
CCListLabels.fold_left2 : f:('a -> 'b -> 'c -> 'a) -> init:'a -> 'b list -> 'c list -> 'a
CCListLabels.fold_left_map : f:('a -> 'b -> 'a * 'c) -> init:'a -> 'b list -> 'a * 'c list
CCListLabels.fold_map : f:('acc -> 'a -> 'acc * 'b) -> init:'acc -> 'a list -> 'acc * 'b list
CCListLabels.fold_map2 : f:('acc -> 'a -> 'b -> 'acc * 'c) -> init:'acc -> 'a list -> 'b list -> 'acc * 'c list
CCListLabels.fold_map_i : f:('acc -> int -> 'a -> 'acc * 'b) -> init:'acc -> 'a list -> 'acc * 'b list
CCListLabels.fold_on_map : f:('a -> 'b) -> reduce:('acc -> 'b -> 'acc) -> init:'acc -> 'a list -> 'acc
CCListLabels.fold_product : f:('c -> 'a -> 'b -> 'c) -> init:'c -> 'a t -> 'b t -> 'c
CCListLabels.fold_right : f:('a -> 'b -> 'b) -> 'a t -> init:'b -> 'b
CCListLabels.fold_right2 : f:('a -> 'b -> 'c -> 'c) -> 'a list -> 'b list -> init:'c -> 'c
CCListLabels.fold_while : f:('a -> 'b -> 'a * [`Continue | `Stop]) -> init:'a -> 'b t -> 'a
CCListLabels.foldi : f:('b -> int -> 'a -> 'b) -> init:'b -> 'a t -> 'b
CCListLabels.foldi2 : f:('c -> int -> 'a -> 'b -> 'c) -> init:'c -> 'a t -> 'b t -> 'c
CCListLabels.for_all : f:('a -> bool) -> 'a list -> bool
CCListLabels.for_all2 : f:('a -> 'b -> bool) -> 'a list -> 'b list -> bool
CCListLabels.get_at_idx : int -> 'a t -> 'a option
CCListLabels.get_at_idx_exn : int -> 'a t -> 'a
CCListLabels.group_by : ?hash:('a -> int) -> ?eq:('a -> 'a -> bool) -> 'a t -> 'a list t
CCListLabels.group_join_by : ?eq:('a -> 'a -> bool) -> ?hash:('a -> int) -> ('b -> 'a) -> 'a t -> 'b t -> ('a * 'b list) t
CCListLabels.group_succ : eq:('a -> 'a -> bool) -> 'a list -> 'a list list
CCListLabels.hd : 'a list -> 'a
CCListLabels.hd_tl : 'a t -> 'a * 'a t
CCListLabels.head_opt : 'a t -> 'a option
CCListLabels.init : int -> f:(int -> 'a) -> 'a t
CCListLabels.insert_at_idx : int -> 'a -> 'a t -> 'a t
CCListLabels.inter : eq:('a -> 'a -> bool) -> 'a t -> 'a t -> 'a t
CCListLabels.interleave : 'a list -> 'a list -> 'a list

Containers
CCListLabels.intersperse : x:'a -> 'a list -> 'a list
CCListLabels.is_empty : 'a t -> bool
CCListLabels.is_sorted : cmp:('a -> 'a -> int) -> 'a list -> bool
CCListLabels.iter : f:('a -> unit) -> 'a list -> unit
CCListLabels.iter2 : f:('a -> 'b -> unit) -> 'a list -> 'b list -> unit
CCListLabels.iteri : f:(int -> 'a -> unit) -> 'a t -> unit
CCListLabels.iteri2 : f:(int -> 'a -> 'b -> unit) -> 'a t -> 'b t -> unit
CCListLabels.join : join_row:('a -> 'b -> 'c option) -> 'a t -> 'b t -> 'c t
CCListLabels.join_all_by : ?eq:('key -> 'key -> bool) -> ?hash:('key -> int) -> ('a -> 'key) -> ('b -> 'key) -> merge:('key -> 'a list -> 'b list -> 'c option) -> 'a t -> 'b t -> 'c t
CCListLabels.join_by : ?eq:('key -> 'key -> bool) -> ?hash:('key -> int) -> ('a -> 'key) -> ('b -> 'key) -> merge:('key -> 'a -> 'b -> 'c option) -> 'a t -> 'b t -> 'c t
CCListLabels.keep_ok : ('a, 'b) result t -> 'a t
CCListLabels.keep_some : 'a option t -> 'a t
CCListLabels.last : int -> 'a t -> 'a t
CCListLabels.last_opt : 'a t -> 'a option
CCListLabels.length : 'a list -> int
CCListLabels.map : f:('a -> 'b) -> 'a t -> 'b t
CCListLabels.map2 : f:('a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list
CCListLabels.map_product_l : f:('a -> 'b list) -> 'a list -> 'b list list
CCListLabels.mapi : f:(int -> 'a -> 'b) -> 'a t -> 'b t
CCListLabels.mem : ?eq:('a -> 'a -> bool) -> 'a -> 'a t -> bool
CCListLabels.mem_assoc : ?eq:('a -> 'a -> bool) -> 'a -> ('a * 'b) t -> bool
CCListLabels.mem_assq : 'a -> map:('a * 'b) list -> bool
CCListLabels.memq : 'a -> set:'a list -> bool
CCListLabels.merge : cmp:('a -> 'a -> int) -> 'a list -> 'a list -> 'a list
CCListLabels.mguard : bool -> unit t
CCListLabels.nth : 'a list -> int -> 'a
CCListLabels.nth_opt : 'a t -> int -> 'a option
CCListLabels.of_gen : 'a gen -> 'a t
CCListLabels.of_iter : 'a iter -> 'a t
CCListLabels.of_seq : 'a Seq.t -> 'a t
CCListLabels.of_seq_rev : 'a Seq.t -> 'a t
CCListLabels.partition : f:('a -> bool) -> 'a list -> 'a list * 'a list
CCListLabels.partition_filter_map : f:('a -> [< `Drop | `Left of 'b | `Right of 'c]) -> 'a list -> 'b list * 'c list
CCListLabels.partition_map : f:('a -> [< `Drop | `Left of 'b | `Right of 'c]) -> 'a list -> 'b list * 'c list

Containers
CCListLabels.partition_map_either : f:('a -> ('b, 'c) CCEither.t) -> 'a list -> 'b list * 'c list
CCListLabels.pp : ?pp_start:unit printer -> ?pp_stop:unit printer -> ?pp_sep:unit printer -> 'a printer -> 'a t printer
CCListLabels.product : f:('a -> 'b -> 'c) -> 'a t -> 'b t -> 'c t
CCListLabels.pure : 'a -> 'a t
CCListLabels.random : 'a random_gen -> 'a t random_gen
CCListLabels.random_choose : 'a t -> 'a random_gen
CCListLabels.random_len : int -> 'a random_gen -> 'a t random_gen
CCListLabels.random_non_empty : 'a random_gen -> 'a t random_gen
CCListLabels.random_sequence : 'a random_gen t -> 'a t random_gen
CCListLabels.range : int -> int -> int t
CCListLabels.range' : int -> int -> int t
CCListLabels.range_by : step:int -> int -> int -> int t
CCListLabels.reduce : f:('a -> 'a -> 'a) -> 'a list -> 'a option
CCListLabels.reduce_exn : f:('a -> 'a -> 'a) -> 'a list -> 'a
CCListLabels.remove : eq:('a -> 'a -> bool) -> key:'a -> 'a t -> 'a t
CCListLabels.remove_assoc : eq:('a -> 'a -> bool) -> 'a -> ('a * 'b) t -> ('a * 'b) t
CCListLabels.remove_assq : 'a -> ('a * 'b) list -> ('a * 'b) list
CCListLabels.remove_at_idx : int -> 'a t -> 'a t
CCListLabels.remove_one : eq:('a -> 'a -> bool) -> 'a -> 'a t -> 'a t
CCListLabels.repeat : int -> 'a t -> 'a t
CCListLabels.replicate : int -> 'a -> 'a t
CCListLabels.return : 'a -> 'a t
CCListLabels.rev : 'a list -> 'a list
CCListLabels.rev_append : 'a list -> 'a list -> 'a list
CCListLabels.rev_map : f:('a -> 'b) -> 'a list -> 'b list
CCListLabels.rev_map2 : f:('a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list
CCListLabels.scan_left : f:('acc -> 'a -> 'acc) -> init:'acc -> 'a list -> 'acc list
CCListLabels.set_at_idx : int -> 'a -> 'a t -> 'a t
CCListLabels.sort : cmp:('a -> 'a -> int) -> 'a list -> 'a list
CCListLabels.sort_uniq : cmp:('a -> 'a -> int) -> 'a list -> 'a list
CCListLabels.sorted_diff : cmp:('a -> 'a -> int) -> 'a list -> 'a list -> 'a list
CCListLabels.sorted_diff_uniq : cmp:('a -> 'a -> int) -> 'a list -> 'a list -> 'a list
CCListLabels.sorted_insert : cmp:('a -> 'a -> int) -> ?uniq:bool -> 'a -> 'a list -> 'a list
CCListLabels.sorted_mem : cmp:('a -> 'a -> int) -> 'a -> 'a list -> bool

Containers
CCListLabels.sorted_merge : cmp:('a -> 'a -> int) -> 'a list -> 'a list -> 'a list
CCListLabels.sorted_merge_uniq : cmp:('a -> 'a -> int) -> 'a list -> 'a list -> 'a list
CCListLabels.sorted_remove : cmp:('a -> 'a -> int) -> ?all:bool -> 'a -> 'a list -> 'a list
CCListLabels.split : ('a * 'b) t -> 'a t * 'b t
CCListLabels.stable_sort : cmp:('a -> 'a -> int) -> 'a list -> 'a list
CCListLabels.sublists_of_len : ?last:('a list -> 'a list option) -> ?offset:int -> len:int -> 'a list -> 'a list list
CCListLabels.subset : eq:('a -> 'a -> bool) -> 'a t -> 'a t -> bool
CCListLabels.tail_opt : 'a t -> 'a t option
CCListLabels.take : int -> 'a t -> 'a t
CCListLabels.take_drop : int -> 'a t -> 'a t * 'a t
CCListLabels.take_drop_while : f:('a -> bool) -> 'a t -> 'a t * 'a t
CCListLabels.take_while : f:('a -> bool) -> 'a t -> 'a t
CCListLabels.tl : 'a list -> 'a list
CCListLabels.to_gen : 'a t -> 'a gen
CCListLabels.to_iter : 'a t -> 'a iter
CCListLabels.to_seq : 'a t -> 'a Seq.t
CCListLabels.to_string : ?start:string -> ?stop:string -> ?sep:string -> ('a -> string) -> 'a t -> string
CCListLabels.union : eq:('a -> 'a -> bool) -> 'a t -> 'a t -> 'a t
CCListLabels.uniq : eq:('a -> 'a -> bool) -> 'a t -> 'a t
CCListLabels.uniq_succ : eq:('a -> 'a -> bool) -> 'a list -> 'a list
CCList.(--) : int -> int -> int t
CCList.(--^) : int -> int -> int t
CCList.(<$>) : ('a -> 'b) -> 'a t -> 'b t
CCList.(<*>) : ('a -> 'b) t -> 'a t -> 'b t
CCList.(>>=) : 'a t -> ('a -> 'b t) -> 'b t
CCList.(>|=) : 'a t -> ('a -> 'b) -> 'b t
CCList.(@) : 'a t -> 'a t -> 'a t
CCList.(and&) : 'a list -> 'b list -> ('a * 'b) list
CCList.(and*) : 'a t -> 'b t -> ('a * 'b) t
CCList.(and+) : 'a t -> 'b t -> ('a * 'b) t
CCList.(let*) : 'a t -> ('a -> 'b t) -> 'b t
CCList.(let+) : 'a t -> ('a -> 'b) -> 'b t
CCList.Assoc.get : eq:('a -> 'a -> bool) -> 'a -> ('a, 'b) t -> 'b option
CCList.Assoc.get_exn : eq:('a -> 'a -> bool) -> 'a -> ('a, 'b) t -> 'b

Containers
CCList.Assoc.keys : ('a, 'b) t -> 'a list
CCList.Assoc.map_values : ('b -> 'c) -> ('a, 'b) t -> ('a, 'c) t
CCList.Assoc.mem : ?eq:('a -> 'a -> bool) -> 'a -> ('a, 'b) t -> bool
CCList.Assoc.remove : eq:('a -> 'a -> bool) -> 'a -> ('a, 'b) t -> ('a, 'b) t
CCList.Assoc.set : eq:('a -> 'a -> bool) -> 'a -> 'b -> ('a, 'b) t -> ('a, 'b) t
CCList.Assoc.update : eq:('a -> 'a -> bool) -> f:('b option -> 'b option) -> 'a -> ('a, 'b) t -> ('a, 'b) t
CCList.Assoc.values : ('a, 'b) t -> 'b list
CCList.Ref.clear : 'a t -> unit
CCList.Ref.create : unit -> 'a t
CCList.Ref.lift : ('a list -> 'b) -> 'a t -> 'b
CCList.Ref.pop : 'a t -> 'a option
CCList.Ref.pop_exn : 'a t -> 'a
CCList.Ref.push : 'a t -> 'a -> unit
CCList.Ref.push_list : 'a t -> 'a list -> unit
CCList.add_nodup : eq:('a -> 'a -> bool) -> 'a -> 'a t -> 'a t
CCList.all_ok : ('a, 'err) result t -> ('a t, 'err) result
CCList.all_some : 'a option t -> 'a t option
CCList.append : 'a t -> 'a t -> 'a t
CCList.assoc : eq:('a -> 'a -> bool) -> 'a -> ('a * 'b) t -> 'b
CCList.assoc_opt : eq:('a -> 'a -> bool) -> 'a -> ('a * 'b) t -> 'b option
CCList.assq : 'a -> ('a * 'b) list -> 'b
CCList.assq_opt : 'a -> ('a * 'b) t -> 'b option
CCList.cartesian_product : 'a t t -> 'a t t
CCList.chunks : int -> 'a list -> 'a list list
CCList.combine : 'a list -> 'b list -> ('a * 'b) list
CCList.combine_gen : 'a list -> 'b list -> ('a * 'b) gen
CCList.combine_shortest : 'a list -> 'b list -> ('a * 'b) list
CCList.compare : ('a -> 'a -> int) -> 'a t -> 'a t -> int
CCList.compare_length_with : 'a t -> int -> int
CCList.compare_lengths : 'a t -> 'b t -> int
CCList.concat : 'a list list -> 'a list
CCList.concat_map : ('a -> 'b list) -> 'a list -> 'b list
CCList.cons : 'a -> 'a list -> 'a list
CCList.cons' : 'a t -> 'a -> 'a t

Containers
CCList.cons_maybe : 'a option -> 'a t -> 'a t
CCList.count : ('a -> bool) -> 'a list -> int
CCList.count_true_false : ('a -> bool) -> 'a list -> int * int
CCList.diagonal : 'a t -> ('a * 'a) t
CCList.drop : int -> 'a t -> 'a t
CCList.drop_while : ('a -> bool) -> 'a t -> 'a t
CCList.empty : 'a t
CCList.equal : ('a -> 'a -> bool) -> 'a t -> 'a t -> bool
CCList.exists : ('a -> bool) -> 'a list -> bool
CCList.exists2 : ('a -> 'b -> bool) -> 'a list -> 'b list -> bool
CCList.fast_sort : ('a -> 'a -> int) -> 'a list -> 'a list
CCList.filter : ('a -> bool) -> 'a t -> 'a t
CCList.filter_map : ('a -> 'b option) -> 'a t -> 'b t
CCList.filteri : (int -> 'a -> bool) -> 'a list -> 'a list
CCList.find : ('a -> bool) -> 'a list -> 'a
CCList.find_all : ('a -> bool) -> 'a list -> 'a list
CCList.find_idx : ('a -> bool) -> 'a t -> (int * 'a) option
CCList.find_map : ('a -> 'b option) -> 'a t -> 'b option
CCList.find_mapi : (int -> 'a -> 'b option) -> 'a t -> 'b option
CCList.find_opt : ('a -> bool) -> 'a t -> 'a option
CCList.find_pred : ('a -> bool) -> 'a t -> 'a option
CCList.find_pred_exn : ('a -> bool) -> 'a t -> 'a
CCList.flat_map : ('a -> 'b t) -> 'a t -> 'b t
CCList.flat_map_i : (int -> 'a -> 'b t) -> 'a t -> 'b t
CCList.flatten : 'a t t -> 'a t
CCList.fold_filter_map : ('acc -> 'a -> 'acc * 'b option) -> 'acc -> 'a list -> 'acc * 'b list
CCList.fold_filter_map_i : ('acc -> int -> 'a -> 'acc * 'b option) -> 'acc -> 'a list -> 'acc * 'b list
CCList.fold_flat_map : ('acc -> 'a -> 'acc * 'b list) -> 'acc -> 'a list -> 'acc * 'b list
CCList.fold_flat_map_i : ('acc -> int -> 'a -> 'acc * 'b list) -> 'acc -> 'a list -> 'acc * 'b list
CCList.fold_left : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a
CCList.fold_left2 : ('a -> 'b -> 'c -> 'a) -> 'a -> 'b list -> 'c list -> 'a
CCList.fold_left_map : ('a -> 'b -> 'a * 'c) -> 'a -> 'b list -> 'a * 'c list
CCList.fold_map : ('acc -> 'a -> 'acc * 'b) -> 'acc -> 'a list -> 'acc * 'b list
CCList.fold_map2 : ('acc -> 'a -> 'b -> 'acc * 'c) -> 'acc -> 'a list -> 'b list -> 'acc * 'c list

Containers
CCList.fold_map_i : ('acc -> int -> 'a -> 'acc * 'b) -> 'acc -> 'a list -> 'acc * 'b list
CCList.fold_on_map : f:('a -> 'b) -> reduce:('acc -> 'b -> 'acc) -> 'acc -> 'a list -> 'acc
CCList.fold_product : ('c -> 'a -> 'b -> 'c) -> 'c -> 'a t -> 'b t -> 'c
CCList.fold_right : ('a -> 'b -> 'b) -> 'a t -> 'b -> 'b
CCList.fold_right2 : ('a -> 'b -> 'c -> 'c) -> 'a list -> 'b list -> 'c -> 'c
CCList.fold_while : ('a -> 'b -> 'a * [`Continue | `Stop]) -> 'a -> 'b t -> 'a
CCList.foldi : ('b -> int -> 'a -> 'b) -> 'b -> 'a t -> 'b
CCList.foldi2 : ('c -> int -> 'a -> 'b -> 'c) -> 'c -> 'a t -> 'b t -> 'c
CCList.for_all : ('a -> bool) -> 'a list -> bool
CCList.for_all2 : ('a -> 'b -> bool) -> 'a list -> 'b list -> bool
CCList.get_at_idx : int -> 'a t -> 'a option
CCList.get_at_idx_exn : int -> 'a t -> 'a
CCList.group_by : ?hash:('a -> int) -> ?eq:('a -> 'a -> bool) -> 'a t -> 'a list t
CCList.group_join_by : ?eq:('a -> 'a -> bool) -> ?hash:('a -> int) -> ('b -> 'a) -> 'a t -> 'b t -> ('a * 'b list) t
CCList.group_succ : eq:('a -> 'a -> bool) -> 'a list -> 'a list list
CCList.hd : 'a list -> 'a
CCList.hd_tl : 'a t -> 'a * 'a t
CCList.head_opt : 'a t -> 'a option
CCList.init : int -> (int -> 'a) -> 'a t
CCList.insert_at_idx : int -> 'a -> 'a t -> 'a t
CCList.inter : eq:('a -> 'a -> bool) -> 'a t -> 'a t -> 'a t
CCList.interleave : 'a list -> 'a list -> 'a list
CCList.intersperse : 'a -> 'a list -> 'a list
CCList.is_empty : 'a t -> bool
CCList.is_sorted : cmp:('a -> 'a -> int) -> 'a list -> bool
CCList.iter : ('a -> unit) -> 'a list -> unit
CCList.iter2 : ('a -> 'b -> unit) -> 'a list -> 'b list -> unit
CCList.iteri : (int -> 'a -> unit) -> 'a t -> unit
CCList.iteri2 : (int -> 'a -> 'b -> unit) -> 'a t -> 'b t -> unit
CCList.join : join_row:('a -> 'b -> 'c option) -> 'a t -> 'b t -> 'c t
CCList.join_all_by : ?eq:('key -> 'key -> bool) -> ?hash:('key -> int) -> ('a -> 'key) -> ('b -> 'key) -> merge:('key -> 'a list -> 'b list -> 'c option) -> 'a t -> 'b t -> 'c t
CCList.join_by : ?eq:('key -> 'key -> bool) -> ?hash:('key -> int) -> ('a -> 'key) -> ('b -> 'key) -> merge:('key -> 'a -> 'b -> 'c option) -> 'a t -> 'b t -> 'c t
CCList.keep_ok : ('a, 'b) result t -> 'a t
CCList.keep_some : 'a option t -> 'a t

Containers
CCList.last : int -> 'a t -> 'a t
CCList.last_opt : 'a t -> 'a option
CCList.length : 'a list -> int
CCList.map : ('a -> 'b) -> 'a t -> 'b t
CCList.map2 : ('a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list
CCList.map_product_l : ('a -> 'b list) -> 'a list -> 'b list list
CCList.mapi : (int -> 'a -> 'b) -> 'a t -> 'b t
CCList.mem : ?eq:('a -> 'a -> bool) -> 'a -> 'a t -> bool
CCList.mem_assoc : ?eq:('a -> 'a -> bool) -> 'a -> ('a * 'b) t -> bool
CCList.mem_assq : 'a -> ('a * 'b) list -> bool
CCList.memq : 'a -> 'a list -> bool
CCList.merge : ('a -> 'a -> int) -> 'a list -> 'a list -> 'a list
CCList.mguard : bool -> unit t
CCList.nth : 'a list -> int -> 'a
CCList.nth_opt : 'a t -> int -> 'a option
CCList.of_gen : 'a gen -> 'a t
CCList.of_iter : 'a iter -> 'a t
CCList.of_seq : 'a Seq.t -> 'a t
CCList.of_seq_rev : 'a Seq.t -> 'a t
CCList.partition : ('a -> bool) -> 'a list -> 'a list * 'a list
CCList.partition_filter_map : ('a -> [< `Drop | `Left of 'b | `Right of 'c]) -> 'a list -> 'b list * 'c list
CCList.partition_map : ('a -> [< `Drop | `Left of 'b | `Right of 'c]) -> 'a list -> 'b list * 'c list
CCList.partition_map_either : ('a -> ('b, 'c) CCEither.t) -> 'a list -> 'b list * 'c list
CCList.pp : ?pp_start:unit printer -> ?pp_stop:unit printer -> ?pp_sep:unit printer -> 'a printer -> 'a t printer
CCList.product : ('a -> 'b -> 'c) -> 'a t -> 'b t -> 'c t
CCList.pure : 'a -> 'a t
CCList.random : 'a random_gen -> 'a t random_gen
CCList.random_choose : 'a t -> 'a random_gen
CCList.random_len : int -> 'a random_gen -> 'a t random_gen
CCList.random_non_empty : 'a random_gen -> 'a t random_gen
CCList.random_sequence : 'a random_gen t -> 'a t random_gen
CCList.range : int -> int -> int t
CCList.range' : int -> int -> int t
CCList.range_by : step:int -> int -> int -> int t

Containers
CCList.reduce : ('a -> 'a -> 'a) -> 'a list -> 'a option
CCList.reduce_exn : ('a -> 'a -> 'a) -> 'a list -> 'a
CCList.remove : eq:('a -> 'a -> bool) -> key:'a -> 'a t -> 'a t
CCList.remove_assoc : eq:('a -> 'a -> bool) -> 'a -> ('a * 'b) t -> ('a * 'b) t
CCList.remove_assq : 'a -> ('a * 'b) list -> ('a * 'b) list
CCList.remove_at_idx : int -> 'a t -> 'a t
CCList.remove_one : eq:('a -> 'a -> bool) -> 'a -> 'a t -> 'a t
CCList.repeat : int -> 'a t -> 'a t
CCList.replicate : int -> 'a -> 'a t
CCList.return : 'a -> 'a t
CCList.rev : 'a list -> 'a list
CCList.rev_append : 'a list -> 'a list -> 'a list
CCList.rev_map : ('a -> 'b) -> 'a list -> 'b list
CCList.rev_map2 : ('a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list
CCList.scan_left : ('acc -> 'a -> 'acc) -> 'acc -> 'a list -> 'acc list
CCList.set_at_idx : int -> 'a -> 'a t -> 'a t
CCList.sort : ('a -> 'a -> int) -> 'a list -> 'a list
CCList.sort_uniq : cmp:('a -> 'a -> int) -> 'a list -> 'a list
CCList.sorted_diff : cmp:('a -> 'a -> int) -> 'a list -> 'a list -> 'a list
CCList.sorted_diff_uniq : cmp:('a -> 'a -> int) -> 'a list -> 'a list -> 'a list
CCList.sorted_insert : cmp:('a -> 'a -> int) -> ?uniq:bool -> 'a -> 'a list -> 'a list
CCList.sorted_mem : cmp:('a -> 'a -> int) -> 'a -> 'a list -> bool
CCList.sorted_merge : cmp:('a -> 'a -> int) -> 'a list -> 'a list -> 'a list
CCList.sorted_merge_uniq : cmp:('a -> 'a -> int) -> 'a list -> 'a list -> 'a list
CCList.sorted_remove : cmp:('a -> 'a -> int) -> ?all:bool -> 'a -> 'a list -> 'a list
CCList.split : ('a * 'b) t -> 'a t * 'b t
CCList.stable_sort : ('a -> 'a -> int) -> 'a list -> 'a list
CCList.sublists_of_len : ?last:('a list -> 'a list option) -> ?offset:int -> int -> 'a list -> 'a list list
CCList.subset : eq:('a -> 'a -> bool) -> 'a t -> 'a t -> bool
CCList.tail_opt : 'a t -> 'a t option
CCList.take : int -> 'a t -> 'a t
CCList.take_drop : int -> 'a t -> 'a t * 'a t
CCList.take_drop_while : ('a -> bool) -> 'a t -> 'a t * 'a t
CCList.take_while : ('a -> bool) -> 'a t -> 'a t

Containers
CCList.tl : 'a list -> 'a list
CCList.to_gen : 'a t -> 'a gen
CCList.to_iter : 'a t -> 'a iter
CCList.to_seq : 'a t -> 'a Seq.t
CCList.to_string : ?start:string -> ?stop:string -> ?sep:string -> ('a -> string) -> 'a t -> string
CCList.union : eq:('a -> 'a -> bool) -> 'a t -> 'a t -> 'a t
CCList.uniq : eq:('a -> 'a -> bool) -> 'a t -> 'a t
CCList.uniq_succ : eq:('a -> 'a -> bool) -> 'a list -> 'a list
CCMap.add : key -> 'a -> 'a t -> 'a t
CCMap.add_iter : 'a t -> (key * 'a) CCMap.iter -> 'a t
CCMap.add_iter_with : f:(key -> 'a -> 'a -> 'a) -> 'a t -> (key * 'a) CCMap.iter -> 'a t
CCMap.add_list : 'a t -> (key * 'a) list -> 'a t
CCMap.add_list_with : f:(key -> 'a -> 'a -> 'a) -> 'a t -> (key * 'a) list -> 'a t
CCMap.add_seq : 'a t -> (key * 'a) Seq.t -> 'a t
CCMap.add_seq_with : f:(key -> 'a -> 'a -> 'a) -> 'a t -> (key * 'a) Seq.t -> 'a t
CCMap.bindings : 'a t -> (key * 'a) list
CCMap.cardinal : 'a t -> int
CCMap.choose : 'a t -> key * 'a
CCMap.choose_opt : 'a t -> (key * 'a) option
CCMap.compare : ('a -> 'a -> int) -> 'a t -> 'a t -> int
CCMap.empty : 'a t
CCMap.equal : ('a -> 'a -> bool) -> 'a t -> 'a t -> bool
CCMap.exists : (key -> 'a -> bool) -> 'a t -> bool
CCMap.filter : (key -> 'a -> bool) -> 'a t -> 'a t
CCMap.filter_map : (key -> 'a -> 'b option) -> 'a t -> 'b t
CCMap.find : key -> 'a t -> 'a
CCMap.find_first : (key -> bool) -> 'a t -> key * 'a
CCMap.find_first_opt : (key -> bool) -> 'a t -> (key * 'a) option
CCMap.find_last : (key -> bool) -> 'a t -> key * 'a
CCMap.find_last_opt : (key -> bool) -> 'a t -> (key * 'a) option
CCMap.find_opt : key -> 'a t -> 'a option
CCMap.fold : (key -> 'a -> 'b -> 'b) -> 'a t -> 'b -> 'b
CCMap.for_all : (key -> 'a -> bool) -> 'a t -> bool
CCMap.get : key -> 'a t -> 'a option

Containers
CCMap.get_or : key -> 'a t -> default:'a -> 'a
CCMap.is_empty : 'a t -> bool
CCMap.iter : (key -> 'a -> unit) -> 'a t -> unit
CCMap.keys : 'a t -> key CCMap.iter
CCMap.map : ('a -> 'b) -> 'a t -> 'b t
CCMap.mapi : (key -> 'a -> 'b) -> 'a t -> 'b t
CCMap.max_binding : 'a t -> key * 'a
CCMap.max_binding_opt : 'a t -> (key * 'a) option
CCMap.mem : key -> 'a t -> bool
CCMap.merge : (key -> 'a option -> 'b option -> 'c option) -> 'a t -> 'b t -> 'c t
CCMap.merge_safe : f:(key -> [`Both of 'a * 'b | `Left of 'a | `Right of 'b] -> 'c option) -> 'a t -> 'b t -> 'c t
CCMap.min_binding : 'a t -> key * 'a
CCMap.min_binding_opt : 'a t -> (key * 'a) option
CCMap.of_iter : (key * 'a) CCMap.iter -> 'a t
CCMap.of_iter_with : f:(key -> 'a -> 'a -> 'a) -> (key * 'a) CCMap.iter -> 'a t
CCMap.of_list : (key * 'a) list -> 'a t
CCMap.of_list_with : f:(key -> 'a -> 'a -> 'a) -> (key * 'a) list -> 'a t
CCMap.of_seq : (key * 'a) Seq.t -> 'a t
CCMap.of_seq_with : f:(key -> 'a -> 'a -> 'a) -> (key * 'a) Seq.t -> 'a t
CCMap.partition : (key -> 'a -> bool) -> 'a t -> 'a t * 'a t
CCMap.pp : ?pp_start:unit CCMap.printer -> ?pp_stop:unit CCMap.printer -> ?pp_arrow:unit CCMap.printer -> ?pp_sep:unit CCMap.printer -> key CCMap.printer -> 'a CCMap.printer -> 'a t CCMap.printer
CCMap.remove : key -> 'a t -> 'a t
CCMap.singleton : key -> 'a -> 'a t
CCMap.split : key -> 'a t -> 'a t * 'a option * 'a t
CCMap.to_iter : 'a t -> (key * 'a) CCMap.iter
CCMap.to_list : 'a t -> (key * 'a) list
CCMap.to_rev_seq : 'a t -> (key * 'a) Seq.t
CCMap.to_seq : 'a t -> (key * 'a) Seq.t
CCMap.to_seq_from : key -> 'a t -> (key * 'a) Seq.t
CCMap.union : (key -> 'a -> 'a -> 'a option) -> 'a t -> 'a t -> 'a t
CCMap.update : key -> ('a option -> 'a option) -> 'a t -> 'a t
CCMap.values : 'a t -> 'a CCMap.iter
CCOption.(<$>) : ('a -> 'b) -> 'a t -> 'b t
CCOption.(<*>) : ('a -> 'b) t -> 'a t -> 'b t

Containers
CCOption.(<+>) : 'a t -> 'a t -> 'a t
CCOption.(>>=) : 'a t -> ('a -> 'b t) -> 'b t
CCOption.(>|=) : 'a t -> ('a -> 'b) -> 'b t
CCOption.(and*) : 'a t -> 'b t -> ('a * 'b) t
CCOption.(and+) : 'a t -> 'b t -> ('a * 'b) t
CCOption.(let*) : 'a t -> ('a -> 'b t) -> 'b t
CCOption.(let+) : 'a t -> ('a -> 'b) -> 'b t
CCOption.bind : 'a t -> ('a -> 'b t) -> 'b t
CCOption.choice : 'a t list -> 'a t
CCOption.choice_iter : 'a t iter -> 'a t
CCOption.choice_seq : 'a t Seq.t -> 'a t
CCOption.compare : ('a -> 'a -> int) -> 'a t -> 'a t -> int
CCOption.equal : ('a -> 'a -> bool) -> 'a t -> 'a t -> bool
CCOption.exists : ('a -> bool) -> 'a t -> bool
CCOption.filter : ('a -> bool) -> 'a t -> 'a t
CCOption.flat_map : ('a -> 'b t) -> 'a t -> 'b t
CCOption.flatten : 'a t t -> 'a t
CCOption.fold : ('a -> 'b -> 'a) -> 'a -> 'b t -> 'a
CCOption.for_all : ('a -> bool) -> 'a t -> bool
CCOption.get_exn : 'a t -> 'a
CCOption.get_exn_or : string -> 'a t -> 'a
CCOption.get_lazy : (unit -> 'a) -> 'a t -> 'a
CCOption.get_or : default:'a -> 'a t -> 'a
CCOption.if_ : ('a -> bool) -> 'a -> 'a option
CCOption.is_none : 'a t -> bool
CCOption.is_some : 'a t -> bool
CCOption.iter : ('a -> unit) -> 'a t -> unit
CCOption.map : ('a -> 'b) -> 'a t -> 'b t
CCOption.map2 : ('a -> 'b -> 'c) -> 'a t -> 'b t -> 'c t
CCOption.map_lazy : (unit -> 'b) -> ('a -> 'b) -> 'a t -> 'b
CCOption.map_or : default:'b -> ('a -> 'b) -> 'a t -> 'b
CCOption.none : 'a t
CCOption.of_list : 'a list -> 'a t
CCOption.of_result : ('a, 'b) result -> 'a t

Containers
CCOption.or_ : else_:'a t -> 'a t -> 'a t
CCOption.or_lazy : else_:(unit -> 'a t) -> 'a t -> 'a t
CCOption.pp : 'a printer -> 'a t printer
CCOption.pure : 'a -> 'a t
CCOption.random : 'a random_gen -> 'a t random_gen
CCOption.return : 'a -> 'a t
CCOption.return_if : bool -> 'a -> 'a t
CCOption.sequence_l : 'a t list -> 'a list t
CCOption.some : 'a -> 'a t
CCOption.to_gen : 'a t -> 'a gen
CCOption.to_iter : 'a t -> 'a iter
CCOption.to_list : 'a t -> 'a list
CCOption.to_result : 'e -> 'a t -> ('a, 'e) result
CCOption.to_result_lazy : (unit -> 'e) -> 'a t -> ('a, 'e) result
CCOption.to_seq : 'a t -> 'a Seq.t
CCOption.value : 'a t -> default:'a -> 'a
CCOption.wrap : ?handler:(exn -> bool) -> ('a -> 'b) -> 'a -> 'b option
CCOption.wrap2 : ?handler:(exn -> bool) -> ('a -> 'b -> 'c) -> 'a -> 'b -> 'c option
CCResult.(<$>) : ('a -> 'b) -> ('a, 'err) t -> ('b, 'err) t
CCResult.(<*>) : ('a -> 'b, 'err) t -> ('a, 'err) t -> ('b, 'err) t
CCResult.(>>=) : ('a, 'err) t -> ('a -> ('b, 'err) t) -> ('b, 'err) t
CCResult.(>|=) : ('a, 'err) t -> ('a -> 'b) -> ('b, 'err) t
CCResult.(and*) : ('a, 'e) t -> ('b, 'e) t -> ('a * 'b, 'e) t
CCResult.(and+) : ('a, 'e) t -> ('b, 'e) t -> ('a * 'b, 'e) t
CCResult.(let*) : ('a, 'e) t -> ('a -> ('b, 'e) t) -> ('b, 'e) t
CCResult.(let+) : ('a, 'e) t -> ('a -> 'b) -> ('b, 'e) t
CCResult.add_ctx : string -> ('a, string) t -> ('a, string) t
CCResult.add_ctxf : ('a, Format.formatter, unit, ('b, string) t -> ('b, string) t) format4 -> 'a
CCResult.both : ('a, 'err) t -> ('b, 'err) t -> ('a * 'b, 'err) t
CCResult.catch : ('a, 'err) t -> ok:('a -> 'b) -> err:('err -> 'b) -> 'b
CCResult.choose : ('a, 'err) t list -> ('a, 'err list) t
CCResult.compare : err:'err ord -> 'a ord -> ('a, 'err) t ord
CCResult.equal : err:'err equal -> 'a equal -> ('a, 'err) t equal
CCResult.fail : 'err -> ('a, 'err) t

Containers
CCResult.fail_fprintf : ('a, Format.formatter, unit, ('b, string) t) format4 -> 'a
CCResult.fail_printf : ('a, Buffer.t, unit, ('b, string) t) format4 -> 'a
CCResult.flat_map : ('a -> ('b, 'err) t) -> ('a, 'err) t -> ('b, 'err) t
CCResult.flatten_l : ('a, 'err) t list -> ('a list, 'err) t
CCResult.fold : ok:('a -> 'b) -> error:('err -> 'b) -> ('a, 'err) t -> 'b
CCResult.fold_iter : ('b -> 'a -> ('b, 'err) t) -> 'b -> 'a iter -> ('b, 'err) t
CCResult.fold_l : ('b -> 'a -> ('b, 'err) t) -> 'b -> 'a list -> ('b, 'err) t
CCResult.fold_ok : ('a -> 'b -> 'a) -> 'a -> ('b, 'c) t -> 'a
CCResult.get_exn : ('a, 'b) t -> 'a
CCResult.get_lazy : ('b -> 'a) -> ('a, 'b) t -> 'a
CCResult.get_or : ('a, 'b) t -> default:'a -> 'a
CCResult.get_or_failwith : ('a, string) t -> 'a
CCResult.guard : (unit -> 'a) -> ('a, exn) t
CCResult.guard_str : (unit -> 'a) -> ('a, string) t
CCResult.guard_str_trace : (unit -> 'a) -> ('a, string) t
CCResult.is_error : ('a, 'err) t -> bool
CCResult.is_ok : ('a, 'err) t -> bool
CCResult.iter : ('a -> unit) -> ('a, 'b) t -> unit
CCResult.iter_err : ('err -> unit) -> ('a, 'err) t -> unit
CCResult.join : (('a, 'err) t, 'err) t -> ('a, 'err) t
CCResult.map : ('a -> 'b) -> ('a, 'err) t -> ('b, 'err) t
CCResult.map2 : ('a -> 'b) -> ('err1 -> 'err2) -> ('a, 'err1) t -> ('b, 'err2) t
CCResult.map_err : ('err1 -> 'err2) -> ('a, 'err1) t -> ('a, 'err2) t
CCResult.map_l : ('a -> ('b, 'err) t) -> 'a list -> ('b list, 'err) t
CCResult.map_or : ('a -> 'b) -> ('a, 'c) t -> default:'b -> 'b
CCResult.of_err : ('a, 'b) error -> ('a, 'b) t
CCResult.of_exn : exn -> ('a, string) t
CCResult.of_exn_trace : exn -> ('a, string) t
CCResult.of_opt : 'a option -> ('a, string) t
CCResult.opt_map : ('a -> ('b, 'c) t) -> 'a option -> ('b option, 'c) t
CCResult.pp : 'a printer -> ('a, string) t printer
CCResult.pp' : 'a printer -> 'e printer -> ('a, 'e) t printer
CCResult.pure : 'a -> ('a, 'err) t
CCResult.retry : int -> (unit -> ('a, 'err) t) -> ('a, 'err list) t

Containers
CCResult.return : 'a -> ('a, 'err) t
CCResult.to_err : ('a, 'b) t -> ('a, 'b) error
CCResult.to_iter : ('a, 'b) t -> 'a iter
CCResult.to_opt : ('a, 'b) t -> 'a option
CCResult.to_seq : ('a, 'b) t -> 'a Seq.t
CCResult.wrap1 : ('a -> 'b) -> 'a -> ('b, exn) t
CCResult.wrap2 : ('a -> 'b -> 'c) -> 'a -> 'b -> ('c, exn) t
CCResult.wrap3 : ('a -> 'b -> 'c -> 'd) -> 'a -> 'b -> 'c -> ('d, exn) t
CCSeq.(--) : int -> int -> int t
CCSeq.(--^) : int -> int -> int t
CCSeq.(<*>) : ('a -> 'b) t -> 'a t -> 'b t
CCSeq.(<.>) : ('a -> 'b) t -> 'a t -> 'b t
CCSeq.(>>-) : 'a t -> ('a -> 'b t) -> 'b t
CCSeq.(>>=) : 'a t -> ('a -> 'b t) -> 'b t
CCSeq.(>|=) : 'a t -> ('a -> 'b) -> 'b t
CCSeq.append : 'a t -> 'a t -> 'a t
CCSeq.compare : 'a ord -> 'a t ord
CCSeq.cons : 'a -> 'a t -> 'a t
CCSeq.cycle : 'a t -> 'a t
CCSeq.drop : int -> 'a t -> 'a t
CCSeq.drop_while : ('a -> bool) -> 'a t -> 'a t
CCSeq.empty : 'a t
CCSeq.equal : 'a equal -> 'a t equal
CCSeq.exists : ('a -> bool) -> 'a t -> bool
CCSeq.exists2 : ('a -> 'b -> bool) -> 'a t -> 'b t -> bool
CCSeq.fair_app : ('a -> 'b) t -> 'a t -> 'b t
CCSeq.fair_flat_map : ('a -> 'b t) -> 'a t -> 'b t
CCSeq.filter : ('a -> bool) -> 'a t -> 'a t
CCSeq.filter_map : ('a -> 'b option) -> 'a t -> 'b t
CCSeq.flat_map : ('a -> 'b t) -> 'a t -> 'b t
CCSeq.flatten : 'a t t -> 'a t
CCSeq.fmap : ('a -> 'b option) -> 'a t -> 'b t
CCSeq.fold : ('a -> 'b -> 'a) -> 'a -> 'b t -> 'a
CCSeq.fold2 : ('acc -> 'a -> 'b -> 'acc) -> 'acc -> 'a t -> 'b t -> 'acc

Containers
CCSeq.fold_left : ('a -> 'b -> 'a) -> 'a -> 'b t -> 'a
CCSeq.for_all : ('a -> bool) -> 'a t -> bool
CCSeq.for_all2 : ('a -> 'b -> bool) -> 'a t -> 'b t -> bool
CCSeq.group : 'a equal -> 'a t -> 'a t t
CCSeq.head : 'a t -> 'a option
CCSeq.head_exn : 'a t -> 'a
CCSeq.interleave : 'a t -> 'a t -> 'a t
CCSeq.is_empty : 'a t -> bool
CCSeq.iter : ('a -> unit) -> 'a t -> unit
CCSeq.iter2 : ('a -> 'b -> unit) -> 'a t -> 'b t -> unit
CCSeq.iteri : (int -> 'a -> unit) -> 'a t -> unit
CCSeq.length : 'a t -> int
CCSeq.map : ('a -> 'b) -> 'a t -> 'b t
CCSeq.map2 : ('a -> 'b -> 'c) -> 'a t -> 'b t -> 'c t
CCSeq.mapi : (int -> 'a -> 'b) -> 'a t -> 'b t
CCSeq.memoize : 'a t -> 'a t
CCSeq.merge : 'a ord -> 'a t -> 'a t -> 'a t
CCSeq.nil : 'a t
CCSeq.of_array : 'a array -> 'a t
CCSeq.of_gen : 'a gen -> 'a t
CCSeq.of_list : 'a list -> 'a t
CCSeq.of_string : string -> char t
CCSeq.pp : ?pp_start:unit printer -> ?pp_stop:unit printer -> ?pp_sep:unit printer -> 'a printer -> 'a t printer
CCSeq.product : 'a t -> 'b t -> ('a * 'b) t
CCSeq.product_with : ('a -> 'b -> 'c) -> 'a t -> 'b t -> 'c t
CCSeq.pure : 'a -> 'a t
CCSeq.range : int -> int -> int t
CCSeq.repeat : ?n:int -> 'a -> 'a t
CCSeq.return : 'a -> 'a t
CCSeq.singleton : 'a -> 'a t
CCSeq.sort : cmp:'a ord -> 'a t -> 'a t
CCSeq.sort_uniq : cmp:'a ord -> 'a t -> 'a t
CCSeq.tail : 'a t -> 'a t option
CCSeq.tail_exn : 'a t -> 'a t

Containers
CCSeq.take : int -> 'a t -> 'a t
CCSeq.take_while : ('a -> bool) -> 'a t -> 'a t
CCSeq.to_array : 'a t -> 'a array
CCSeq.to_gen : 'a t -> 'a gen
CCSeq.to_iter : 'a t -> 'a iter
CCSeq.to_list : 'a t -> 'a list
CCSeq.to_rev_list : 'a t -> 'a list
CCSeq.unfold : ('b -> ('a * 'b) option) -> 'b -> 'a t
CCSeq.uniq : 'a equal -> 'a t -> 'a t
CCSeq.unzip : ('a * 'b) t -> 'a t * 'b t
CCSeq.zip : 'a t -> 'b t -> ('a * 'b) t
CCSeq.zip_i : 'a t -> (int * 'a) t
CCSet.add : elt -> t -> t
CCSet.add_iter : t -> elt iter -> t
CCSet.add_list : t -> elt list -> t
CCSet.add_seq : elt Seq.t -> t -> t
CCSet.cardinal : t -> int
CCSet.choose : t -> elt
CCSet.choose_opt : t -> elt option
CCSet.compare : t -> t -> int
CCSet.diff : t -> t -> t
CCSet.disjoint : t -> t -> bool
CCSet.elements : t -> elt list
CCSet.empty : t
CCSet.equal : t -> t -> bool
CCSet.exists : (elt -> bool) -> t -> bool
CCSet.filter : (elt -> bool) -> t -> t
CCSet.filter_map : (elt -> elt option) -> t -> t
CCSet.find : elt -> t -> elt
CCSet.find_first : (elt -> bool) -> t -> elt
CCSet.find_first_opt : (elt -> bool) -> t -> elt option
CCSet.find_last : (elt -> bool) -> t -> elt
CCSet.find_last_opt : (elt -> bool) -> t -> elt option
CCSet.find_opt : elt -> t -> elt option

Containers
CCSet.fold : (elt -> 'a -> 'a) -> t -> 'a -> 'a
CCSet.for_all : (elt -> bool) -> t -> bool
CCSet.inter : t -> t -> t
CCSet.is_empty : t -> bool
CCSet.iter : (elt -> unit) -> t -> unit
CCSet.map : (elt -> elt) -> t -> t
CCSet.max_elt : t -> elt
CCSet.max_elt_opt : t -> elt option
CCSet.mem : elt -> t -> bool
CCSet.min_elt : t -> elt
CCSet.min_elt_opt : t -> elt option
CCSet.of_iter : elt iter -> t
CCSet.of_list : elt list -> t
CCSet.of_seq : elt Seq.t -> t
CCSet.partition : (elt -> bool) -> t -> t * t
CCSet.pp : ?pp_start:unit printer -> ?pp_stop:unit printer -> ?pp_sep:unit printer -> elt printer -> t printer
CCSet.remove : elt -> t -> t
CCSet.singleton : elt -> t
CCSet.split : elt -> t -> t * bool * t
CCSet.subset : t -> t -> bool
CCSet.to_iter : t -> elt iter
CCSet.to_list : t -> elt list
CCSet.to_rev_seq : t -> elt Seq.t
CCSet.to_seq : t -> elt Seq.t
CCSet.to_seq_from : elt -> t -> elt Seq.t
CCSet.to_string : ?start:string -> ?stop:string -> ?sep:string -> (elt -> string) -> t -> string
CCSet.union : t -> t -> t
CCStringLabels.(<) : t -> t -> bool
CCStringLabels.(<=) : t -> t -> bool
CCStringLabels.(<>) : t -> t -> bool
CCStringLabels.(=) : t -> t -> bool
CCStringLabels.(>) : t -> t -> bool
CCStringLabels.(>=) : t -> t -> bool
CCStringLabels.blit : src:t -> src_pos:int -> dst:Bytes.t -> dst_pos:int -> len:int -> unit

Containers
CCStringLabels.capitalize : string -> string
CCStringLabels.capitalize_ascii : string -> string
CCStringLabels.cat : string -> string -> string
CCStringLabels.chop_prefix : pre:string -> string -> string option
CCStringLabels.chop_suffix : suf:string -> string -> string option
CCStringLabels.compare : string -> string -> int
CCStringLabels.compare_natural : string -> string -> int
CCStringLabels.compare_versions : string -> string -> int
CCStringLabels.concat : sep:string -> string list -> string
CCStringLabels.concat_gen : sep:string -> string gen -> string
CCStringLabels.concat_iter : sep:string -> string iter -> string
CCStringLabels.concat_seq : sep:string -> string Seq.t -> string
CCStringLabels.contains : string -> char -> bool
CCStringLabels.contains_from : string -> int -> char -> bool
CCStringLabels.copy : string -> string
CCStringLabels.drop : int -> string -> string
CCStringLabels.drop_while : f:(char -> bool) -> t -> t
CCStringLabels.edit_distance : ?cutoff:int -> string -> string -> int
CCStringLabels.empty : string
CCStringLabels.ends_with : suffix:string -> string -> bool
CCStringLabels.equal : string -> string -> bool
CCStringLabels.equal_caseless : string -> string -> bool
CCStringLabels.escaped : string -> string
CCStringLabels.exists : f:(char -> bool) -> string -> bool
CCStringLabels.exists2 : f:(char -> char -> bool) -> string -> string -> bool
CCStringLabels.fill : bytes -> pos:int -> len:int -> char -> unit
CCStringLabels.filter : f:(char -> bool) -> string -> string
CCStringLabels.filter_map : f:(char -> char option) -> string -> string
CCStringLabels.find : ?start:int -> sub:string -> string -> int
CCStringLabels.find_all : ?start:int -> sub:string -> string -> int gen
CCStringLabels.find_all_l : ?start:int -> sub:string -> string -> int list
CCStringLabels.flat_map : ?sep:string -> f:(char -> string) -> string -> string
CCStringLabels.fold : f:('a -> char -> 'a) -> init:'a -> t -> 'a
CCStringLabels.fold2 : f:('a -> char -> char -> 'a) -> init:'a -> string -> string -> 'a

Containers
CCStringLabels.fold_left : f:('a -> char -> 'a) -> init:'a -> string -> 'a
CCStringLabels.fold_right : f:(char -> 'a -> 'a) -> string -> init:'a -> 'a
CCStringLabels.foldi : f:('a -> int -> char -> 'a) -> 'a -> t -> 'a
CCStringLabels.for_all : f:(char -> bool) -> string -> bool
CCStringLabels.for_all2 : f:(char -> char -> bool) -> string -> string -> bool
CCStringLabels.get_int16_be : string -> int -> int
CCStringLabels.get_int16_le : string -> int -> int
CCStringLabels.get_int16_ne : string -> int -> int
CCStringLabels.get_int32_be : string -> int -> int32
CCStringLabels.get_int32_le : string -> int -> int32
CCStringLabels.get_int32_ne : string -> int -> int32
CCStringLabels.get_int64_be : string -> int -> int64
CCStringLabels.get_int64_le : string -> int -> int64
CCStringLabels.get_int64_ne : string -> int -> int64
CCStringLabels.get_int8 : string -> int -> int
CCStringLabels.get_uint16_be : string -> int -> int
CCStringLabels.get_uint16_le : string -> int -> int
CCStringLabels.get_uint16_ne : string -> int -> int
CCStringLabels.get_uint8 : string -> int -> int
CCStringLabels.get_utf_16be_uchar : t -> int -> Uchar.utf_decode
CCStringLabels.get_utf_16le_uchar : t -> int -> Uchar.utf_decode
CCStringLabels.get_utf_8_uchar : t -> int -> Uchar.utf_decode
CCStringLabels.hash : string -> int
CCStringLabels.index : string -> char -> int
CCStringLabels.index_from : string -> int -> char -> int
CCStringLabels.index_from_opt : string -> int -> char -> int option
CCStringLabels.index_opt : string -> char -> int option
CCStringLabels.init : int -> f:(int -> char) -> string
CCStringLabels.is_empty : string -> bool
CCStringLabels.is_sub : sub:string -> sub_pos:int -> string -> pos:int -> sub_len:int -> bool
CCStringLabels.is_valid_utf_16be : t -> bool
CCStringLabels.is_valid_utf_16le : t -> bool
CCStringLabels.is_valid_utf_8 : t -> bool
CCStringLabels.iter : f:(char -> unit) -> string -> unit

Containers
CCStringLabels.iter2 : f:(char -> char -> unit) -> string -> string -> unit
CCStringLabels.iteri : f:(int -> char -> unit) -> string -> unit
CCStringLabels.iteri2 : f:(int -> char -> char -> unit) -> string -> string -> unit
CCStringLabels.length : t -> int
CCStringLabels.lines : string -> string list
CCStringLabels.lines_gen : string -> string gen
CCStringLabels.lines_iter : string -> string iter
CCStringLabels.lines_seq : string -> string Seq.t
CCStringLabels.lowercase : string -> string
CCStringLabels.lowercase_ascii : string -> string
CCStringLabels.ltrim : t -> t
CCStringLabels.make : int -> char -> string
CCStringLabels.map : f:(char -> char) -> string -> string
CCStringLabels.map2 : f:(char -> char -> char) -> string -> string -> string
CCStringLabels.mapi : f:(int -> char -> char) -> string -> string
CCStringLabels.mem : ?start:int -> sub:string -> string -> bool
CCStringLabels.of_array : char array -> string
CCStringLabels.of_bytes : bytes -> string
CCStringLabels.of_char : char -> string
CCStringLabels.of_gen : char gen -> string
CCStringLabels.of_hex : string -> string option
CCStringLabels.of_hex_exn : string -> string
CCStringLabels.of_iter : char iter -> string
CCStringLabels.of_list : char list -> string
CCStringLabels.of_seq : char Seq.t -> string
CCStringLabels.pad : ?side:[`Left | `Right] -> ?c:char -> int -> string -> string
CCStringLabels.pp : Format.formatter -> t -> unit
CCStringLabels.pp_buf : Buffer.t -> t -> unit
CCStringLabels.prefix : pre:string -> string -> bool
CCStringLabels.rcontains_from : string -> int -> char -> bool
CCStringLabels.rdrop_while : f:(char -> bool) -> t -> t
CCStringLabels.repeat : string -> int -> string
CCStringLabels.replace : ?which:[`All | `Left | `Right] -> sub:string -> by:string -> string -> string
CCStringLabels.rev : string -> string

Containers
CCStringLabels.rfind : sub:string -> string -> int
CCStringLabels.rindex : string -> char -> int
CCStringLabels.rindex_from : string -> int -> char -> int
CCStringLabels.rindex_from_opt : string -> int -> char -> int option
CCStringLabels.rindex_opt : string -> char -> int option
CCStringLabels.rtrim : t -> t
CCStringLabels.set : string -> int -> char -> string
CCStringLabels.split : by:string -> string -> string list
CCStringLabels.split_on_char : by:char -> string -> string list
CCStringLabels.starts_with : prefix:string -> string -> bool
CCStringLabels.sub : string -> pos:int -> len:int -> string
CCStringLabels.suffix : suf:string -> string -> bool
CCStringLabels.take : int -> string -> string
CCStringLabels.take_drop : int -> string -> string * string
CCStringLabels.to_array : string -> char array
CCStringLabels.to_bytes : string -> bytes
CCStringLabels.to_gen : t -> char gen
CCStringLabels.to_hex : string -> string
CCStringLabels.to_iter : t -> char iter
CCStringLabels.to_list : t -> char list
CCStringLabels.to_seq : t -> char Seq.t
CCStringLabels.to_seqi : t -> (int * char) Seq.t
CCStringLabels.trim : string -> string
CCStringLabels.uncapitalize : string -> string
CCStringLabels.uncapitalize_ascii : string -> string
CCStringLabels.uniq : eq:(char -> char -> bool) -> string -> string
CCStringLabels.unlines : string list -> string
CCStringLabels.unlines_gen : string gen -> string
CCStringLabels.unlines_iter : string iter -> string
CCStringLabels.unlines_seq : string Seq.t -> string
CCStringLabels.uppercase : string -> string
CCStringLabels.uppercase_ascii : string -> string
CCString.(<) : t -> t -> bool
CCString.(<=) : t -> t -> bool

Containers
CCString.(<>) : t -> t -> bool
CCString.(=) : t -> t -> bool
CCString.(>) : t -> t -> bool
CCString.(>=) : t -> t -> bool
CCString.blit : t -> int -> Bytes.t -> int -> int -> unit
CCString.capitalize : string -> string
CCString.capitalize_ascii : string -> string
CCString.cat : string -> string -> string
CCString.chop_prefix : pre:string -> string -> string option
CCString.chop_suffix : suf:string -> string -> string option
CCString.compare : string -> string -> int
CCString.compare_natural : string -> string -> int
CCString.compare_versions : string -> string -> int
CCString.concat : string -> string list -> string
CCString.concat_gen : sep:string -> string gen -> string
CCString.concat_iter : sep:string -> string iter -> string
CCString.concat_seq : sep:string -> string Seq.t -> string
CCString.contains : string -> char -> bool
CCString.contains_from : string -> int -> char -> bool
CCString.copy : string -> string
CCString.drop : int -> string -> string
CCString.drop_while : (char -> bool) -> t -> t
CCString.edit_distance : ?cutoff:int -> string -> string -> int
CCString.empty : string
CCString.ends_with : suffix:string -> string -> bool
CCString.equal : t -> t -> bool
CCString.equal_caseless : string -> string -> bool
CCString.escaped : string -> string
CCString.exists : (char -> bool) -> string -> bool
CCString.exists2 : (char -> char -> bool) -> string -> string -> bool
CCString.fill : bytes -> int -> int -> char -> unit
CCString.filter : (char -> bool) -> string -> string
CCString.filter_map : (char -> char option) -> string -> string
CCString.find : ?start:int -> sub:string -> string -> int

Containers
CCString.find_all : ?start:int -> sub:string -> string -> int gen
CCString.find_all_l : ?start:int -> sub:string -> string -> int list
CCString.flat_map : ?sep:string -> (char -> string) -> string -> string
CCString.fold : ('a -> char -> 'a) -> 'a -> t -> 'a
CCString.fold2 : ('a -> char -> char -> 'a) -> 'a -> string -> string -> 'a
CCString.fold_left : ('a -> char -> 'a) -> 'a -> string -> 'a
CCString.fold_right : (char -> 'a -> 'a) -> string -> 'a -> 'a
CCString.foldi : ('a -> int -> char -> 'a) -> 'a -> t -> 'a
CCString.for_all : (char -> bool) -> string -> bool
CCString.for_all2 : (char -> char -> bool) -> string -> string -> bool
CCString.get_int16_be : string -> int -> int
CCString.get_int16_le : string -> int -> int
CCString.get_int16_ne : string -> int -> int
CCString.get_int32_be : string -> int -> int32
CCString.get_int32_le : string -> int -> int32
CCString.get_int32_ne : string -> int -> int32
CCString.get_int64_be : string -> int -> int64
CCString.get_int64_le : string -> int -> int64
CCString.get_int64_ne : string -> int -> int64
CCString.get_int8 : string -> int -> int
CCString.get_uint16_be : string -> int -> int
CCString.get_uint16_le : string -> int -> int
CCString.get_uint16_ne : string -> int -> int
CCString.get_uint8 : string -> int -> int
CCString.get_utf_16be_uchar : t -> int -> Uchar.utf_decode
CCString.get_utf_16le_uchar : t -> int -> Uchar.utf_decode
CCString.get_utf_8_uchar : t -> int -> Uchar.utf_decode
CCString.hash : string -> int
CCString.index : string -> char -> int
CCString.index_from : string -> int -> char -> int
CCString.index_from_opt : string -> int -> char -> int option
CCString.index_opt : string -> char -> int option
CCString.init : int -> (int -> char) -> string
CCString.is_empty : string -> bool

Containers
CCString.is_sub : sub:string -> int -> string -> int -> sub_len:int -> bool
CCString.is_valid_utf_16be : t -> bool
CCString.is_valid_utf_16le : t -> bool
CCString.is_valid_utf_8 : t -> bool
CCString.iter : (char -> unit) -> string -> unit
CCString.iter2 : (char -> char -> unit) -> string -> string -> unit
CCString.iteri : (int -> char -> unit) -> string -> unit
CCString.iteri2 : (int -> char -> char -> unit) -> string -> string -> unit
CCString.length : t -> int
CCString.lines : string -> string list
CCString.lines_gen : string -> string gen
CCString.lines_iter : string -> string iter
CCString.lines_seq : string -> string Seq.t
CCString.lowercase : string -> string
CCString.lowercase_ascii : string -> string
CCString.ltrim : t -> t
CCString.make : int -> char -> string
CCString.map : (char -> char) -> string -> string
CCString.map2 : (char -> char -> char) -> string -> string -> string
CCString.mapi : (int -> char -> char) -> string -> string
CCString.mem : ?start:int -> sub:string -> string -> bool
CCString.of_array : char array -> string
CCString.of_bytes : bytes -> string
CCString.of_char : char -> string
CCString.of_gen : char gen -> string
CCString.of_hex : string -> string option
CCString.of_hex_exn : string -> string
CCString.of_iter : char iter -> string
CCString.of_list : char list -> string
CCString.of_seq : char Seq.t -> string
CCString.pad : ?side:[`Left | `Right] -> ?c:char -> int -> string -> string
CCString.pp : Format.formatter -> t -> unit
CCString.pp_buf : Buffer.t -> t -> unit
CCString.prefix : pre:string -> string -> bool

Containers
CCString.rcontains_from : string -> int -> char -> bool
CCString.rdrop_while : (char -> bool) -> t -> t
CCString.repeat : string -> int -> string
CCString.replace : ?which:[`All | `Left | `Right] -> sub:string -> by:string -> string -> string
CCString.rev : string -> string
CCString.rfind : sub:string -> string -> int
CCString.rindex : string -> char -> int
CCString.rindex_from : string -> int -> char -> int
CCString.rindex_from_opt : string -> int -> char -> int option
CCString.rindex_opt : string -> char -> int option
CCString.rtrim : t -> t
CCString.set : string -> int -> char -> string
CCString.split : by:string -> string -> string list
CCString.split_on_char : char -> string -> string list
CCString.starts_with : prefix:string -> string -> bool
CCString.sub : string -> int -> int -> string
CCString.suffix : suf:string -> string -> bool
CCString.take : int -> string -> string
CCString.take_drop : int -> string -> string * string
CCString.to_array : string -> char array
CCString.to_bytes : string -> bytes
CCString.to_gen : t -> char gen
CCString.to_hex : string -> string
CCString.to_iter : t -> char iter
CCString.to_list : t -> char list
CCString.to_seq : t -> char Seq.t
CCString.to_seqi : t -> (int * char) Seq.t
CCString.trim : string -> string
CCString.uncapitalize : string -> string
CCString.uncapitalize_ascii : string -> string
CCString.uniq : (char -> char -> bool) -> string -> string
CCString.unlines : string list -> string
CCString.unlines_gen : string gen -> string
CCString.unlines_iter : string iter -> string

Containers
CCString.unlines_seq : string Seq.t -> string
CCString.uppercase : string -> string
CCString.uppercase_ascii : string -> string