Containers
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.bsearch : cmp:('a -> 'a -> int) -> 'a -> 'a t -> [`All_bigger | `All_lower | `At of int | `Empty | `Just_after of int]
CCArray.compare : 'a ord -> 'a t ord
CCArray.empty : 'a t
CCArray.equal : 'a equal -> 'a t equal
CCArray.except_idx : 'a t -> int -> 'a list
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_i : (int -> 'a -> 'b option) -> 'a t -> 'b option
CCArray.flat_map : ('a -> 'b t) -> 'a t -> 'b array
CCArray.fold : ('a -> 'b -> 'a) -> 'a -> 'b t -> 'a
CCArray.fold_map : ('acc -> 'a -> 'acc * 'b) -> 'acc -> 'a t -> 'acc * 'b t
CCArray.fold_while : ('a -> 'b -> 'a * [`Continue | `Stop]) -> 'a -> 'b t -> 'a
CCArray.fold2 : ('acc -> 'a -> 'b -> 'acc) -> 'acc -> 'a t -> 'b t -> 'acc
CCArray.foldi : ('a -> int -> 'b -> 'a) -> 'a -> 'b t -> 'a
CCArray.get_safe : 'a t -> int -> 'a option
CCArray.lookup : cmp:'a ord -> 'a -> 'a t -> int option
CCArray.lookup_exn : cmp:'a ord -> 'a -> 'a t -> int
CCArray.map_inplace : ('a -> 'a) -> 'a array -> unit
CCArray.monoid_product : ('a -> 'b -> 'c) -> 'a t -> 'b t -> 'c t
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

Containers
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_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.swap : 'a t -> int -> int -> unit
CCArray.to_gen : 'a t -> 'a gen
CCArray.to_iter : 'a t -> 'a iter
CCArray.to_string : ?sep:string -> ('a -> string) -> 'a array -> string
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.bsearch : cmp:('a -> 'a -> int) -> key:'a -> 'a t -> [`All_bigger | `All_lower | `At of int | `Empty | `Just_after of int]
CCArrayLabels.compare : 'a ord -> 'a t ord
CCArrayLabels.empty : 'a t
CCArrayLabels.equal : 'a equal -> 'a t equal
CCArrayLabels.except_idx : 'a t -> int -> 'a list
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_i : f:(int -> 'a -> 'b option) -> 'a t -> 'b option
CCArrayLabels.flat_map : f:('a -> 'b t) -> 'a t -> 'b array
CCArrayLabels.fold : f:('a -> 'b -> 'a) -> init:'a -> 'b t -> 'a

Containers
CCArrayLabels.fold_map : f:('acc -> 'a -> 'acc * 'b) -> init:'acc -> 'a t -> 'acc * 'b t
CCArrayLabels.fold2 : f:('acc -> 'a -> 'b -> 'acc) -> init:'acc -> 'a t -> 'b t -> 'acc
CCArrayLabels.foldi : f:('a -> int -> 'b -> 'a) -> init:'a -> 'b t -> 'a
CCArrayLabels.get_safe : 'a t -> int -> 'a option
CCArrayLabels.lookup : cmp:'a ord -> key:'a -> 'a t -> int option
CCArrayLabels.lookup_exn : cmp:'a ord -> key:'a -> 'a t -> int
CCArrayLabels.map_inplace : f:('a -> 'a) -> 'a t -> unit
CCArrayLabels.monoid_product : f:('a -> 'b -> 'c) -> 'a t -> 'b t -> 'c t
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
CCArrayLabels.shuffle : 'a t -> unit
CCArrayLabels.shuffle_with : Random.State.t -> 'a t -> 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.swap : 'a t -> int -> int -> unit
CCArrayLabels.to_gen : 'a t -> 'a gen
CCArrayLabels.to_iter : 'a t -> 'a iter
CCArrayLabels.to_string : ?sep:string -> ('a -> string) -> 'a array -> string
CCList.(--) : int -> int -> int t
CCList.(--^) : int -> int -> int t
CCList.(@) : 'a t -> 'a t -> 'a t
CCList.(<*>) : ('a -> 'b) t -> 'a t -> 'b t
CCList.(<$>) : ('a -> 'b) -> 'a t -> 'b t
CCList.(>>=) : 'a t -> ('a -> 'b t) -> 'b t
CCList.(>|=) : 'a t -> ('a -> 'b) -> 'b t

Containers
CCList.(and*) : 'a t -> 'b t -> ('a * 'b) t
CCList.(and&) : 'a list -> 'b list -> ('a * 'b) list
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.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.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
CCList.Assoc.keys : ('a, 'b) t -> 'a list
CCList.Assoc.map_values : ('b -> 'c) -> ('a, 'b) t -> ('a
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.cartesian_product : 'a t t -> 'a t t
CCList.chunks : int -> 'a list -> 'a list list
CCList.combine_gen : 'a list -> 'b list -> ('a * 'b) gen
CCList.combine_shortest : 'a list -> 'b list -> ('a * 'b) list
CCList.cons_maybe : 'a option -> 'a t -> 'a t
CCList.cons' : 'a t -> 'a -> '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.find_idx : ('a -> bool) -> 'a t -> (int * 'a) option
CCList.find_mapi : (int -> 'a -> 'b option) -> 'a t -> 'b 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

Containers
CCList.flat_map_i : (int -> 'a -> 'b t) -> 'a t -> 'b 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_map : ('acc -> 'a -> 'acc * 'b) -> 'acc -> 'a list -> 'acc * 'b list
CCList.fold_map_i : ('acc -> int -> '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
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_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.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_tl : 'a t -> 'a * 'a t
CCList.head_opt : 'a t -> 'a option
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.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
CCList.last : int -> 'a t -> 'a t
CCList.last_opt : 'a t -> 'a option

Containers
CCList.map_product_l : ('a -> 'b list) -> 'a list -> 'b list list
CCList.mguard : bool -> unit t
CCList.of_gen : 'a gen -> 'a t
CCList.of_iter : 'a iter -> 'a t
CCList.of_seq_rev : 'a Seq.t -> 'a t
CCList.partition_filter_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_by : step:int -> int -> int -> int t
CCList.range' : int -> int -> int t
CCList.reduce : ('a -> 'a -> 'a) -> 'a list -> 'a option
CCList.reduce_exn : ('a -> 'a -> 'a) -> 'a list -> 'a
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.remove : eq:('a -> 'a -> bool) -> key:'a -> 'a t -> 'a t
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.scan_left : ('acc -> 'a -> 'acc) -> 'acc -> 'a list -> 'acc list

Containers
CCList.set_at_idx : int -> 'a -> 'a t -> 'a t
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.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
CCList.to_gen : 'a t -> 'a gen
CCList.to_iter : 'a t -> 'a iter
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
CCListLabels.(--) : int -> int -> int CCList.t
CCListLabels.(--^) : int -> int -> int CCList.t
CCListLabels.(@) : 'a CCList.t -> 'a CCList.t -> 'a CCList.t
CCListLabels.(<*>) : ('a -> 'b) CCList.t -> 'a CCList.t -> 'b CCList.t
CCListLabels.(<$>) : ('a -> 'b) -> '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.(and*) : 'a CCList.t -> 'b CCList.t -> ('a * 'b) CCList.t
CCListLabels.(and&) : 'a list -> 'b list -> ('a * 'b) list
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.add_nodup : eq:('a -> 'a -> bool) -> 'a -> 'a t -> 'a t

Containers
CCListLabels.all_ok : ('a, 'err) result t -> ('a t, 'err) result
CCListLabels.all_some : 'a option t -> 'a t option
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.mem : ?eq:('a -> 'a -> bool) -> 'a -> ('a, 'b) t -> bool
CCListLabels.Assoc.remove : eq:('a -> 'a -> bool) -> 'a -> ('a, 'b) t -> ('a, 'b) 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.cartesian_product : 'a t t -> 'a t t
CCListLabels.chunks : int -> 'a list -> 'a list list
CCListLabels.combine_gen : 'a list -> 'b list -> ('a * 'b) gen
CCListLabels.combine_shortest : 'a list -> 'b list -> ('a * 'b) list
CCListLabels.cons_maybe : 'a option -> 'a t -> 'a t
CCListLabels.cons' : 'a t -> 'a -> '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.find_idx : f:('a -> bool) -> 'a t -> (int * 'a) option
CCListLabels.find_mapi : f:(int -> 'a -> 'b option) -> 'a t -> 'b option
CCListLabels.find_pred : f:('a -> bool) -> 'a t -> 'a option
CCListLabels.find_pred_exn : f:('a -> bool) -> 'a t -> 'a
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.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_map : f:('acc -> 'a -> 'acc * 'b) -> init:'acc -> 'a list -> 'acc * 'b list

Containers
CCListLabels.fold_map_i : f:('acc -> int -> '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_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_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.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_tl : 'a t -> 'a * 'a t
CCListLabels.head_opt : 'a t -> 'a option
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
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.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.map_product_l : f:('a -> 'b list) -> 'a list -> 'b list list
CCListLabels.mguard : bool -> unit t
CCListLabels.of_gen : 'a gen -> 'a t
CCListLabels.of_iter : 'a iter -> 'a t
CCListLabels.of_seq_rev : 'a Seq.t -> 'a t
CCListLabels.partition_filter_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_by : step:int -> int -> int -> int t
CCListLabels.range' : 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.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.remove : eq:('a -> 'a -> bool) -> key:'a -> 'a t -> 'a t
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.scan_left : f:('acc -> 'a -> 'acc) -> init:'acc -> 'a list -> 'acc list
CCListLabels.set_at_idx : int -> 'a -> 'a t -> 'a t
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
CCListLabels.sorted_merge : cmp:('a -> 'a -> int) -> 'a list -> 'a list -> 'a list

Containers
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.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.to_gen : 'a t -> 'a gen
CCListLabels.to_iter : 'a t -> 'a iter
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
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_with : f:(key -> 'a -> 'a -> 'a) -> 'a t -> (key * 'a) Seq.t -> 'a t
CCMap.get : key -> 'a t -> 'a option
CCMap.get_or : key -> 'a t -> default:'a -> 'a
CCMap.keys : 'a t -> key CCMap.iter
CCMap.merge_safe : f:(key -> [`Both of 'a * 'b | `Left of 'a | `Right of 'b] -> 'c option) -> 'a t -> 'b t -> 'c t
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_with : f:(key -> 'a -> 'a -> 'a) -> (key * 'a) Seq.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.to_iter : 'a t -> (key * 'a) CCMap.iter
CCMap.to_list : 'a t -> (key * 'a) list
CCMap.values : 'a t -> 'a CCMap.iter
CCOption.(<*>) : ('a -> 'b) t -> 'a t -> 'b t

Containers
CCOption.(<+>) : 'a t -> 'a t -> 'a t
CCOption.(<$>) : ('a -> 'b) -> 'a t -> 'b 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.choice : 'a t list -> 'a t
CCOption.choice_iter : 'a t iter -> 'a t
CCOption.choice_seq : 'a t Seq.t -> 'a t
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.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.map_lazy : (unit -> 'b) -> ('a -> 'b) -> 'a t -> 'b
CCOption.map_or : default:'b -> ('a -> 'b) -> 'a t -> 'b
CCOption.map2 : ('a -> 'b -> 'c) -> 'a t -> 'b t -> 'c t
CCOption.of_list : 'a list -> 'a t
CCOption.of_result : ('a, 'b) result -> 'a t
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

Containers
CCOption.to_gen : 'a t -> 'a gen
CCOption.to_iter : 'a t -> 'a iter
CCOption.to_result_lazy : (unit -> 'e) -> 'a t -> ('a, 'e) result
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, 'err) t -> ('a, 'err) t -> ('b, 'err) t
CCResult.(<$>) : ('a -> 'b) -> ('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.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_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.map_l : ('a -> ('b, 'err) t) -> 'a list -> ('b list, 'err) t
CCResult.map_or : ('a -> 'b) -> ('a, 'c) t -> default:'b -> 'b

Containers
CCResult.map2 : ('a -> 'b) -> ('err1 -> 'err2) -> ('a, 'err1) t -> ('b, 'err2) t
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
CCResult.return : 'a -> ('a, 'err) t
CCResult.to_err : ('a, 'b) t -> ('a, 'b) error
CCResult.to_iter : ('a, 'b) t -> 'a iter
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.fair_app : ('a -> 'b) t -> 'a t -> 'b t
CCSeq.fair_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_left : ('a -> 'b -> 'a) -> 'a -> 'b t -> 'a
CCSeq.head : 'a t -> 'a option
CCSeq.head_exn : 'a t -> 'a
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

Containers
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_with : ('a -> 'b -> 'c) -> 'a t -> 'b t -> 'c t
CCSeq.pure : 'a -> 'a t
CCSeq.range : int -> int -> int 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
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.uniq : 'a equal -> 'a t -> 'a t
CCSeq.zip_i : 'a t -> (int * 'a) t
CCSet.add_iter : t -> elt iter -> t
CCSet.add_list : t -> elt list -> t
CCSet.of_iter : elt iter -> t
CCSet.pp : ?pp_start:unit printer -> ?pp_stop:unit printer -> ?pp_sep:unit printer -> elt printer -> t printer
CCSet.to_iter : t -> elt iter
CCSet.to_list : t -> elt list
CCSet.to_string : ?start:string -> ?stop:string -> ?sep:string -> (elt -> string) -> t -> string
CCString.(<) : t -> t -> bool
CCString.(<=) : t -> t -> bool
CCString.(<>) : t -> t -> bool
CCString.(=) : t -> t -> bool
CCString.(>) : t -> t -> bool
CCString.(>=) : t -> t -> bool
CCString.chop_prefix : pre:string -> string -> string option
CCString.chop_suffix : suf:string -> string -> string option
CCString.compare_natural : string -> string -> int

Containers
CCString.compare_versions : string -> string -> int
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.drop : int -> string -> string
CCString.drop_while : (char -> bool) -> t -> t
CCString.edit_distance : ?cutoff:int -> string -> string -> int
CCString.equal_caseless : string -> string -> bool
CCString.exists2 : (char -> char -> bool) -> string -> string -> bool
CCString.filter : (char -> bool) -> string -> string
CCString.filter_map : (char -> char option) -> string -> string
CCString.find : ?start:int -> sub:string -> string -> int
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.foldi : ('a -> int -> char -> 'a) -> 'a -> t -> 'a
CCString.for_all2 : (char -> char -> bool) -> string -> string -> bool
CCString.hash : string -> int
CCString.is_empty : string -> bool
CCString.is_sub : sub:string -> int -> string -> int -> sub_len:int -> bool
CCString.iter2 : (char -> char -> unit) -> string -> 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.ltrim : t -> t
CCString.map2 : (char -> char -> char) -> string -> string -> string
CCString.mem : ?start:int -> sub:string -> string -> bool
CCString.of_array : char array -> string
CCString.of_char : char -> string

Containers
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.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
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.rtrim : t -> t
CCString.set : string -> int -> char -> string
CCString.split : by:string -> string -> string list
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_gen : t -> char gen
CCString.to_hex : string -> string
CCString.to_iter : t -> char iter
CCString.to_list : t -> char list
CCString.uniq : (char -> char -> bool) -> string -> string
CCString.unlines : string list -> string
CCString.unlines_gen : string gen -> string
CCString.unlines_iter : string iter -> string
CCString.unlines_seq : string Seq.t -> string
CCStringLabels.(<) : t -> t -> bool
CCStringLabels.(<=) : t -> t -> bool
CCStringLabels.(<>) : t -> t -> bool
CCStringLabels.(=) : t -> t -> bool

Containers
CCStringLabels.(>) : t -> t -> bool
CCStringLabels.(>=) : t -> t -> bool
CCStringLabels.chop_prefix : pre:string -> string -> string option
CCStringLabels.chop_suffix : suf:string -> string -> string option
CCStringLabels.compare_natural : string -> string -> int
CCStringLabels.compare_versions : string -> string -> int
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.drop : int -> string -> string
CCStringLabels.drop_while : f:(char -> bool) -> t -> t
CCStringLabels.edit_distance : ?cutoff:int -> string -> string -> int
CCStringLabels.equal_caseless : string -> string -> bool
CCStringLabels.exists2 : f:(char -> char -> bool) -> string -> string -> bool
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
CCStringLabels.foldi : f:('a -> int -> char -> 'a) -> 'a -> t -> 'a
CCStringLabels.for_all2 : f:(char -> char -> bool) -> string -> string -> bool
CCStringLabels.hash : string -> int
CCStringLabels.is_empty : string -> bool
CCStringLabels.is_sub : sub:string -> sub_pos:int -> string -> pos:int -> sub_len:int -> bool
CCStringLabels.iter2 : f:(char -> char -> unit) -> string -> 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

Containers
CCStringLabels.ltrim : t -> t
CCStringLabels.map2 : f:(char -> char -> char) -> string -> string -> string
CCStringLabels.mem : ?start:int -> sub:string -> string -> bool
CCStringLabels.of_array : char array -> 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.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.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
CCStringLabels.rfind : sub:string -> string -> int
CCStringLabels.rtrim : t -> t
CCStringLabels.set : string -> int -> char -> string
CCStringLabels.split : by:string -> string -> string list
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_gen : t -> char gen
CCStringLabels.to_hex : string -> string
CCStringLabels.to_iter : t -> char iter
CCStringLabels.to_list : t -> char list
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

Containers
CCStringLabels.unlines_seq : string Seq.t -> string