Containers	Batteries	Base
CCArrayLabels.filter : f:('a -> bool) -> 'a array -> 'a array	BatArray.Labels.filter : f:('a -> bool) -> 'a array -> 'a array	Base.Array.filter : 'a array -> f:('a -> bool) -> 'a array
CCArrayLabels.filter_map : f:('a -> 'b option) -> 'a array -> 'b array	BatArray.Labels.filter_map : f:('a -> 'b option) -> 'a array -> 'b array	Base.Array.filter_map : 'a array -> f:('a -> 'b option) -> 'b array
CCArrayLabels.fold : f:('a -> 'b -> 'a) -> init:'a -> 'b array -> 'a	BatArray.Labels.fold : f:('a -> 'b -> 'a) -> init:'a -> 'b array -> 'a	Base.Array.fold : 'a array -> init:'accum -> f:('accum -> 'a -> 'accum) -> 'accum
CCListLabels.cartesian_product : 'a list list -> 'a list list	BatList.n_cartesian_product : 'a list list -> 'a list list	Base.List.all : 'a list list -> 'a list list
CCListLabels.count : f:('a -> bool) -> 'a list -> int	BatList.Labels.count_matching : f:('a -> bool) -> 'a list -> int	Base.List.count : 'a list -> f:('a -> bool) -> int
CCListLabels.drop : int -> 'a list -> 'a list	BatList.drop : int -> 'a list -> 'a list	Base.List.drop : 'a list -> int -> 'a list
CCListLabels.drop_while : f:('a -> bool) -> 'a list -> 'a list	BatList.Labels.drop_while : f:('a -> bool) -> 'a list -> 'a list	Base.List.drop_while : 'a list -> f:('a -> bool) -> 'a list
CCListLabels.take : int -> 'a list -> 'a list	BatList.take : int -> 'a list -> 'a list	Base.List.take : 'a list -> int -> 'a list
CCListLabels.take_while : f:('a -> bool) -> 'a list -> 'a list	BatList.Labels.take_while : f:('a -> bool) -> 'a list -> 'a list	Base.List.take_while : 'a list -> f:('a -> bool) -> 'a list
CCMap.Make.keys	BatMap.keys : ('a, 'b) map -> 'a BatEnum.t	Base.Map.keys : ('k, 'a, 'b) map -> 'k list
CCOpt.filter : ('a -> bool) -> 'a option -> 'a option	BatOption.filter : ('a -> bool) -> 'a option -> 'a option	Base.Option.filter : 'a option -> f:('a -> bool) -> 'a option
CCSeq.drop : int -> 'a Seq.t -> 'a Seq.t	BatSeq.drop : int -> 'a Seq.t -> 'a Seq.t	Base.Sequence.drop : 'a Base.Sequence.t -> int -> 'a Base.Sequence.t
CCSeq.drop_while : ('a -> bool) -> 'a Seq.t -> 'a Seq.t	BatSeq.drop_while : ('a -> bool) -> 'a Seq.t -> 'a Seq.t	Base.Sequence.drop_while : 'a Base.Sequence.t -> f:('a -> bool) -> 'a Base.Sequence.t
CCSeq.equal : 'a CCSeq.equal -> 'a Seq.t CCSeq.equal	BatSeq.equal : ?eq:('a -> 'a -> bool) -> 'a Seq.t -> 'a Seq.t -> bool	Base.Sequence.equal : ('a -> 'a -> bool) -> 'a Base.Sequence.t -> 'a Base.Sequence.t -> bool
CCSeq.exists : ('a -> bool) -> 'a Seq.t -> bool	BatSeq.exists : ('a -> bool) -> 'a Seq.t -> bool	Base.Sequence.exists : 'a Base.Sequence.t -> f:('a -> bool) -> bool
CCSeq.for_all : ('a -> bool) -> 'a Seq.t -> bool	BatSeq.for_all : ('a -> bool) -> 'a Seq.t -> bool	Base.Sequence.for_all : 'a Base.Sequence.t -> f:('a -> bool) -> bool
CCSeq.head_exn : 'a Seq.t -> 'a	BatSeq.hd : 'a Seq.t -> 'a	Base.Sequence.hd_exn : 'a Base.Sequence.t -> 'a
CCSeq.is_empty : 'a Seq.t -> bool	BatSeq.is_empty : 'a Seq.t -> bool	Base.Sequence.is_empty : 'a Base.Sequence.t -> bool
CCSeq.iteri : (int -> 'a -> unit) -> 'a Seq.t -> unit	BatSeq.iteri : (int -> 'a -> unit) -> 'a Seq.t -> unit	Base.Sequence.iteri : ('a Base.Sequence.t, 'a) Base.Indexed_container_intf.iteri
CCSeq.length : 'a Seq.t -> int	BatSeq.length : 'a Seq.t -> int	Base.Sequence.length : 'a Base.Sequence.t -> int
CCSeq.mapi : (int -> 'a -> 'b) -> 'a Seq.t -> 'b Seq.t	BatSeq.mapi : (int -> 'a -> 'b) -> 'a Seq.t -> 'b Seq.t	Base.Sequence.mapi : 'a Base.Sequence.t -> f:(int -> 'a -> 'b) -> 'b Base.Sequence.t
CCSeq.of_list : 'a list -> 'a Seq.t	BatSeq.of_list : 'a list -> 'a Seq.t	Base.Sequence.of_list : 'a list -> 'a Base.Sequence.t
CCSeq.take : int -> 'a Seq.t -> 'a Seq.t	BatSeq.take : int -> 'a Seq.t -> 'a Seq.t	Base.Sequence.take : 'a Base.Sequence.t -> int -> 'a Base.Sequence.t
CCSeq.take_while : ('a -> bool) -> 'a Seq.t -> 'a Seq.t	BatSeq.take_while : ('a -> bool) -> 'a Seq.t -> 'a Seq.t	Base.Sequence.take_while : 'a Base.Sequence.t -> f:('a -> bool) -> 'a Base.Sequence.t
CCSet.Make.to_list	BatSet.to_list : 'a set -> 'a list	Base.Set.to_list : ('a, 'b) set -> 'a list
CCStringLabels.filter : f:(char -> bool) -> string -> string	BatString.filter : (char -> bool) -> string -> string	Base.String.filter : string -> f:(char -> bool) -> string
CCStringLabels.fold : f:('a -> char -> 'a) -> init:'a -> string -> 'a	BatString.fold_left : ('a -> char -> 'a) -> 'a -> string -> 'a	Base.String.fold : string -> init:'accum -> f:('accum -> Base.String.elt -> 'accum) -> 'accum
CCStringLabels.is_empty : string -> bool	BatString.is_empty : string -> bool	Base.String.is_empty : string -> bool
CCStringLabels.of_char : char -> string	BatString.of_char : char -> string	Base.String.of_char : char -> string
CCStringLabels.rev : string -> string	BatString.rev : string -> string	Base.String.rev : string -> string
CCStringLabels.to_list : string -> char list	BatString.to_list : string -> char list	Base.String.to_list : string -> Base.String.elt list