Stdlib	Containers	Batteries	Base
		BatMap.(-->) : ('a, 'b) map -> 'a -> 'b	
		BatMap.(<--) : ('a, 'b) map -> 'a * 'b -> ('a, 'b) map	
Map.Make.add	CCMap.Make.add	BatMap.add : 'a -> 'b -> ('a, 'b) map -> ('a, 'b) map	Base.Map.add : ('k, 'v, 'cmp) map -> key:'k -> data:'v -> ('k, 'v, 'cmp) map Base.Map.Or_duplicate.t
		BatMap.add_carry : 'a -> 'b -> ('a, 'b) map -> ('a, 'b) map * 'b option	
			Base.Map.add_exn : ('k, 'v, 'cmp) map -> key:'k -> data:'v -> ('k, 'v, 'cmp) map
	CCMap.Make.add_iter		
	CCMap.Make.add_iter_with		
	CCMap.Make.add_list		
	CCMap.Make.add_list_with		
			Base.Map.add_multi : ('k, 'v list, 'cmp) map -> key:'k -> data:'v -> ('k, 'v list, 'cmp) map
Map.Make.add_seq	CCMap.Make.add_seq	BatMap.add_seq : ('key * 'a) BatSeq.t -> ('key, 'a) map -> ('key, 'a) map	
	CCMap.Make.add_seq_with		
		BatMap.any : ('key, 'a) map -> 'key * 'a	
			Base.Map.append : lower_part:('k, 'v, 'cmp) map -> upper_part:('k, 'v, 'cmp) map -> [`Ok of ('k, 'v, 'cmp) map | `Overlapping_key_ranges]
		BatMap.at_rank_exn : int -> ('key, 'a) map -> 'key * 'a	
		BatMap.backwards : ('a, 'b) map -> ('a * 'b) BatEnum.t	
			Base.Map.binary_search : ('k, 'v, 'cmp) map -> compare:(key:'k -> data:'v -> 'key -> int) -> [`First_equal_to | `First_greater_than_or_equal_to | `First_strictly_greater_than | `Last_equal_to | `Last_less_than_or_equal_to | `Last_strictly_less_than] -> 'key -> ('k * 'v) option
			Base.Map.binary_search_segmented : ('k, 'v, 'cmp) map -> segment_of:(key:'k -> data:'v -> [`Left | `Right]) -> [`First_on_right | `Last_on_left] -> ('k * 'v) option
Map.Make.bindings	CCMap.Make.bindings	BatMap.bindings : ('key, 'a) map -> ('key * 'a) list	
Map.Make.cardinal	CCMap.Make.cardinal	BatMap.cardinal : ('a, 'b) map -> int	
			Base.Map.change : ('k, 'v, 'cmp) map -> 'k -> f:('v option -> 'v option) -> ('k, 'v, 'cmp) map
Map.Make.choose	CCMap.Make.choose	BatMap.choose : ('key, 'a) map -> 'key * 'a	
Map.Make.choose_opt	CCMap.Make.choose_opt	BatMap.choose_opt : ('key, 'a) map -> ('key * 'a) option	
			Base.Map.closest_key : ('k, 'v, 'cmp) map -> [`Greater_or_equal_to | `Greater_than | `Less_or_equal_to | `Less_than] -> 'k -> ('k * 'v) option
			Base.Map.combine_errors : ('k, 'v Base__.Or_error.t, 'cmp) map -> ('k, 'v, 'cmp) map Base__.Or_error.t
			Base.Map.comparator : ('a, 'b, 'cmp) map -> ('a, 'cmp) Base__.Comparator.t
			Base.Map.comparator_s : ('a, 'b, 'cmp) map -> ('a, 'cmp) Base.Map.comparator
Map.Make.compare	CCMap.Make.compare	BatMap.compare : ('b -> 'b -> int) -> ('a, 'b) map -> ('a, 'b) map -> int	
			Base.Map.compare_direct : ('v -> 'v -> int) -> ('k, 'v, 'cmp) map -> ('k, 'v, 'cmp) map -> int
			Base.Map.compare_m__t : (module Base.Map.Compare_m) -> ('v -> 'v -> int) -> ('k, 'v, 'cmp) map -> ('k, 'v, 'cmp) map -> int
			Base.Map.count : ('k, 'v, 'a) map -> f:('v -> bool) -> int
			Base.Map.counti : ('k, 'v, 'a) map -> f:(key:'k -> data:'v -> bool) -> int
			Base.Map.data : ('a, 'v, 'b) map -> 'v list
		BatMap.diff : ('a, 'b) map -> ('a, 'b) map -> ('a, 'b) map	
Map.Make.empty	CCMap.Make.empty	BatMap.empty : ('a, 'b) map	Base.Map.empty : ('a, 'cmp) Base.Map.comparator -> ('a, 'b, 'cmp) map
		BatMap.enum : ('a, 'b) map -> ('a * 'b) BatEnum.t	
Map.Make.equal	CCMap.Make.equal	BatMap.equal : ('b -> 'b -> bool) -> ('a, 'b) map -> ('a, 'b) map -> bool	Base.Map.equal : ('v -> 'v -> bool) -> ('k, 'v, 'cmp) map -> ('k, 'v, 'cmp) map -> bool
			Base.Map.equal_m__t : (module Base.Map.Equal_m) -> ('v -> 'v -> bool) -> ('k, 'v, 'cmp) map -> ('k, 'v, 'cmp) map -> bool

Stdlib	Containers	Batteries	Base
Map.Make.exists	CCMap.Make.exists	BatMap.exists : ('a -> 'b -> bool) -> ('a, 'b) map -> bool	Base.Map.exists : ('k, 'v, 'a) map -> f:('v -> bool) -> bool
			Base.Map.existsi : ('k, 'v, 'a) map -> f:(key:'k -> data:'v -> bool) -> bool
		BatMap.extract : 'a -> ('a, 'b) map -> 'b * ('a, 'b) map	
Map.Make.filter	CCMap.Make.filter	BatMap.filter : ('key -> 'a -> bool) -> ('key, 'a) map -> ('key, 'a) map	Base.Map.filter : ('k, 'v, 'cmp) map -> f:('v -> bool) -> ('k, 'v, 'cmp) map
			Base.Map.filter_keys : ('k, 'v, 'cmp) map -> f:('k -> bool) -> ('k, 'v, 'cmp) map
Map.Make.filter_map	CCMap.Make.filter_map	BatMap.filter_map : ('key -> 'a -> 'b option) -> ('key, 'a) map -> ('key, 'b) map	Base.Map.filter_map : ('k, 'v1, 'cmp) map -> f:('v1 -> 'v2 option) -> ('k, 'v2, 'cmp) map
			Base.Map.filter_mapi : ('k, 'v1, 'cmp) map -> f:(key:'k -> data:'v1 -> 'v2 option) -> ('k, 'v2, 'cmp) map
			Base.Map.filteri : ('k, 'v, 'cmp) map -> f:(key:'k -> data:'v -> bool) -> ('k, 'v, 'cmp) map
		BatMap.filterv : ('a -> bool) -> ('key, 'a) map -> ('key, 'a) map	
Map.Make.find	CCMap.Make.find	BatMap.find : 'a -> ('a, 'b) map -> 'b	Base.Map.find : ('k, 'v, 'cmp) map -> 'k -> 'v option
		BatMap.find_default : 'b -> 'a -> ('a, 'b) map -> 'b	
			Base.Map.find_exn : ('k, 'v, 'cmp) map -> 'k -> 'v
Map.Make.find_first	CCMap.Make.find_first	BatMap.find_first : ('a -> bool) -> ('a, 'b) map -> 'a * 'b	
Map.Make.find_first_opt	CCMap.Make.find_first_opt	BatMap.find_first_opt : ('a -> bool) -> ('a, 'b) map -> ('a * 'b) option	
Map.Make.find_last	CCMap.Make.find_last	BatMap.find_last : ('a -> bool) -> ('a, 'b) map -> 'a * 'b	
Map.Make.find_last_opt	CCMap.Make.find_last_opt	BatMap.find_last_opt : ('a -> bool) -> ('a, 'b) map -> ('a * 'b) option	
			Base.Map.find_multi : ('k, 'v list, 'cmp) map -> 'k -> 'v list
Map.Make.find_opt	CCMap.Make.find_opt	BatMap.find_opt : 'a -> ('a, 'b) map -> 'b option	
Map.Make.fold	CCMap.Make.fold	BatMap.fold : ('b -> 'c -> 'c) -> ('a, 'b) map -> 'c -> 'c	Base.Map.fold : ('k, 'v, 'b) map -> init:'a -> f:(key:'k -> data:'v -> 'a -> 'a) -> 'a
			Base.Map.fold2 : ('k, 'v1, 'cmp) map -> ('k, 'v2, 'cmp) map -> init:'a -> f:(key:'k -> data:[`Both of 'v1 * 'v2 | `Left of 'v1 | `Right of 'v2] -> 'a -> 'a) -> 'a
		BatMap.foldi : ('a -> 'b -> 'c -> 'c) -> ('a, 'b) map -> 'c -> 'c	
			Base.Map.fold_range_inclusive : ('k, 'v, 'cmp) map -> min:'k -> max:'k -> init:'a -> f:(key:'k -> data:'v -> 'a -> 'a) -> 'a
			Base.Map.fold_right : ('k, 'v, 'b) map -> init:'a -> f:(key:'k -> data:'v -> 'a -> 'a) -> 'a
			Base.Map.fold_symmetric_diff : ('k, 'v, 'cmp) map -> ('k, 'v, 'cmp) map -> data_equal:('v -> 'v -> bool) -> init:'a -> f:('a -> ('k, 'v) Base.Map.Symmetric_diff_element.t -> 'a) -> 'a
Map.Make.for_all	CCMap.Make.for_all	BatMap.for_all : ('a -> 'b -> bool) -> ('a, 'b) map -> bool	Base.Map.for_all : ('k, 'v, 'a) map -> f:('v -> bool) -> bool
			Base.Map.for_alli : ('k, 'v, 'a) map -> f:(key:'k -> data:'v -> bool) -> bool
	CCMap.Make.get		
	CCMap.Make.get_or		
			Base.Map.hash_fold_direct : 'k Base__.Hash.folder -> 'v Base__.Hash.folder -> ('k, 'v, 'cmp) map Base__.Hash.folder
			Base.Map.hash_fold_m__t : (module Base.Map.Hash_fold_m with type t = 'k) -> (Base__.Hash.state -> 'v -> Base__.Hash.state) -> Base__.Hash.state -> ('k, 'v, 'a) map -> Base__.Hash.state
		BatMap.intersect : ('b -> 'c -> 'd) -> ('a, 'b) map -> ('a, 'c) map -> ('a, 'd) map	
			Base.Map.invariants : ('a, 'b, 'c) map -> bool
Map.Make.is_empty	CCMap.Make.is_empty	BatMap.is_empty : ('a, 'b) map -> bool	Base.Map.is_empty : ('a, 'b, 'c) map -> bool
Map.Make.iter	CCMap.Make.iter	BatMap.iter : ('a -> 'b -> unit) -> ('a, 'b) map -> unit	Base.Map.iter : ('a, 'v, 'b) map -> f:('v -> unit) -> unit
			Base.Map.iter2 : ('k, 'v1, 'cmp) map -> ('k, 'v2, 'cmp) map -> f:(key:'k -> data:[`Both of 'v1 * 'v2 | `Left of 'v1 | `Right of 'v2] -> unit) -> unit
			Base.Map.iter_keys : ('k, 'a, 'b) map -> f:('k -> unit) -> unit
			Base.Map.iteri : ('k, 'v, 'a) map -> f:(key:'k -> data:'v -> unit) -> unit
			Base.Map.iteri_until : ('k, 'v, 'a) map -> f:(key:'k -> data:'v -> Base.Map.Continue_or_stop.t) -> Base.Map.Finished_or_unfinished.t
	CCMap.Make.keys	BatMap.keys : ('a, 'b) map -> 'a BatEnum.t	Base.Map.keys : ('k, 'a, 'b) map -> 'k list

Stdlib	Containers	Batteries	Base
			Base.Map.length : ('a, 'b, 'c) map -> int
			Base.Map.m__t_of_sexp : (module Base.Map.M_of_sexp with type comparator_witness = 'cmp and type t = 'k) -> (Base__.Sexp.t -> 'v) -> Base__.Sexp.t -> ('k, 'v, 'cmp) map
			Base.Map.m__t_sexp_grammar : Base__.Ppx_sexp_conv_lib.Sexp.Private.Raw_grammar.t ...)
Map.Make.map	CCMap.Make.map	BatMap.map : ('b -> 'c) -> ('a, 'b) map -> ('a, 'c) map	Base.Map.map : ('k, 'v1, 'cmp) map -> f:('v1 -> 'v2) -> ('k, 'v2, 'cmp) map
Map.Make.mapi	CCMap.Make.mapi	BatMap.mapi : ('a -> 'b -> 'c) -> ('a, 'b) map -> ('a, 'c) map	Base.Map.mapi : ('k, 'v1, 'cmp) map -> f:(key:'k -> data:'v1 -> 'v2) -> ('k, 'v2, 'cmp) map
Map.Make.max_binding	CCMap.Make.max_binding	BatMap.max_binding : ('key, 'a) map -> 'key * 'a	
Map.Make.max_binding_opt	CCMap.Make.max_binding_opt	BatMap.max_binding_opt : ('key, 'a) map -> ('key * 'a) option	
			Base.Map.max_elt : ('k, 'v, 'a) map -> ('k * 'v) option
			Base.Map.max_elt_exn : ('k, 'v, 'a) map -> 'k * 'v
Map.Make.mem	CCMap.Make.mem	BatMap.mem : 'a -> ('a, 'b) map -> bool	Base.Map.mem : ('k, 'a, 'cmp) map -> 'k -> bool
Map.Make.merge	CCMap.Make.merge	BatMap.merge : ('key -> 'a option -> 'b option -> 'c option) -> ('key, 'a) map -> ('key, 'b) map -> ('key, 'c) map	Base.Map.merge : ('k, 'v1, 'cmp) map -> ('k, 'v2, 'cmp) map -> f:(key:'k -> [`Both of 'v1 * 'v2 | `Left of 'v1 | `Right of 'v2] -> 'v3 option) -> ('k, 'v3, 'cmp) map
	CCMap.Make.merge_safe		
			Base.Map.merge_skewed : ('k, 'v, 'cmp) map -> ('k, 'v, 'cmp) map -> combine:(key:'k -> 'v -> 'v -> 'v) -> ('k, 'v, 'cmp) map
Map.Make.min_binding	CCMap.Make.min_binding	BatMap.min_binding : ('key, 'a) map -> 'key * 'a	
Map.Make.min_binding_opt	CCMap.Make.min_binding_opt	BatMap.min_binding_opt : ('key, 'a) map -> ('key * 'a) option	
			Base.Map.min_elt : ('k, 'v, 'a) map -> ('k * 'v) option
			Base.Map.min_elt_exn : ('k, 'v, 'a) map -> 'k * 'v
		BatMap.modify : 'a -> ('b -> 'b) -> ('a, 'b) map -> ('a, 'b) map	
		BatMap.modify_def : 'b -> 'a -> ('b -> 'b) -> ('a, 'b) map -> ('a, 'b) map	
		BatMap.modify_opt : 'a -> ('b option -> 'b option) -> ('a, 'b) map -> ('a, 'b) map	
			Base.Map.nth : ('k, 'v, 'a) map -> int -> ('k * 'v) option
			Base.Map.nth_exn : ('k, 'v, 'a) map -> int -> 'k * 'v
			Base.Map.of_alist : ('a, 'cmp) Base.Map.comparator -> ('a * 'b) list -> [`Duplicate_key of 'a | `Ok of ('a, 'b, 'cmp) map]
			Base.Map.of_alist_exn : ('a, 'cmp) Base.Map.comparator -> ('a * 'b) list -> ('a, 'b, 'cmp) map
			Base.Map.of_alist_fold : ('a, 'cmp) Base.Map.comparator -> ('a * 'b) list -> init:'c -> f:('c -> 'b -> 'c) -> ('a, 'c, 'cmp) map
			Base.Map.of_alist_multi : ('a, 'cmp) Base.Map.comparator -> ('a * 'b) list -> ('a, 'b list, 'cmp) map
			Base.Map.of_alist_or_error : ('a, 'cmp) Base.Map.comparator -> ('a * 'b) list -> ('a, 'b, 'cmp) map Base__.Or_error.t
			Base.Map.of_alist_reduce : ('a, 'cmp) Base.Map.comparator -> ('a * 'b) list -> f:('b -> 'b -> 'b) -> ('a, 'b, 'cmp) map
		BatMap.of_enum : ('a * 'b) BatEnum.t -> ('a, 'b) map	
			Base.Map.of_increasing_iterator_unchecked : ('a, 'cmp) Base.Map.comparator -> len:int -> f:(int -> 'a * 'b) -> ('a, 'b, 'cmp) map
			Base.Map.of_increasing_sequence : ('k, 'cmp) Base.Map.comparator -> ('k * 'v) Base__.Sequence.t -> ('k, 'v, 'cmp) map Base__.Or_error.t
	CCMap.Make.of_iter		
	CCMap.Make.of_iter_with		
	CCMap.Make.of_list		Base.Map.of_iteri : ('a, 'cmp) Base.Map.comparator -> iteri:(f:(key:'a -> data:'b -> unit) -> unit) -> [`Duplicate_key of 'a | `Ok of ('a, 'b, 'cmp) map]
	CCMap.Make.of_list_with		
Map.Make.of_seq	CCMap.Make.of_seq	BatMap.of_seq : ('key * 'a) BatSeq.t -> ('key, 'a) map	
	CCMap.Make.of_seq_with		
			Base.Map.of_sequence : ('k, 'cmp) Base.Map.comparator -> ('k * 'v) Base__.Sequence.t -> [`Duplicate_key of 'k | `Ok of ('k, 'v, 'cmp) map]

Stdlib	Containers	Batteries	Base
			Base.Map.of_sequence_exn : ('a, 'cmp) Base.Map.comparator -> ('a * 'b) Base__.Sequence.t -> ('a, 'b, 'cmp) map
			Base.Map.of_sequence_fold : ('a, 'cmp) Base.Map.comparator -> ('a * 'b) Base__.Sequence.t -> init:'c -> f:('c -> 'b -> 'c) -> ('a, 'c, 'cmp) map
			Base.Map.of_sequence_multi : ('a, 'cmp) Base.Map.comparator -> ('a * 'b) Base__.Sequence.t -> ('a, 'b list, 'cmp) map
			Base.Map.of_sequence_or_error : ('a, 'cmp) Base.Map.comparator -> ('a * 'b) Base__.Sequence.t -> ('a, 'b, 'cmp) map Base__.Or_error.t
			Base.Map.of_sequence_reduce : ('a, 'cmp) Base.Map.comparator -> ('a * 'b) Base__.Sequence.t -> f:('b -> 'b -> 'b) -> ('a, 'b, 'cmp) map
			Base.Map.of_sorted_array : ('a, 'cmp) Base.Map.comparator -> ('a * 'b) array -> ('a, 'b, 'cmp) map Base__.Or_error.t
			Base.Map.of_sorted_array_unchecked : ('a, 'cmp) Base.Map.comparator -> ('a * 'b) array -> ('a, 'b, 'cmp) map
Map.Make.partition	CCMap.Make.partition	BatMap.partition : ('a -> 'b -> bool) -> ('a, 'b) map -> ('a, 'b) map * ('a, 'b) map	
			Base.Map.partition_map : ('k, 'v1, 'cmp) map -> f:('v1 -> ('v2, 'v3) Base__.Either.t) -> ('k, 'v2, 'cmp) map * ('k, 'v3, 'cmp) map
			Base.Map.partition_mapi : ('k, 'v1, 'cmp) map -> f:(key:'k -> data:'v1 -> ('v2, 'v3) Base__.Either.t) -> ('k, 'v2, 'cmp) map * ('k, 'v3, 'cmp) map
			Base.Map.partition_tf : ('k, 'v, 'cmp) map -> f:('v -> bool) -> ('k, 'v, 'cmp) map * ('k, 'v, 'cmp) map
			Base.Map.partitioni_tf : ('k, 'v, 'cmp) map -> f:(key:'k -> data:'v -> bool) -> ('k, 'v, 'cmp) map * ('k, 'v, 'cmp) map
		BatMap.pop : ('a, 'b) map -> ('a * 'b) * ('a, 'b) map	
		BatMap.pop_max_binding : ('key, 'a) map -> ('key * 'a) * ('key, 'a) map	
		BatMap.pop_min_binding : ('key, 'a) map -> ('key * 'a) * ('key, 'a) map	
	CCMap.Make.pp		
		BatMap.print : ?first:string -> ?last:string -> ?sep:string -> ?kvsep:string -> ('a BatInnerIO.output -> 'b -> unit) -> ('a BatInnerIO.output -> 'c -> unit) -> 'a BatInnerIO.output -> ('b, 'c) map -> unit	
			Base.Map.range_to_alist : ('k, 'v, 'cmp) map -> min:'k -> max:'k -> ('k * 'v) list
			Base.Map.rank : ('k, 'v, 'cmp) map -> 'k -> int option
Map.Make.remove	CCMap.Make.remove	BatMap.remove : 'a -> ('a, 'b) map -> ('a, 'b) map	Base.Map.remove : ('k, 'v, 'cmp) map -> 'k -> ('k, 'v, 'cmp) map
		BatMap.remove_exn : 'a -> ('a, 'b) map -> ('a, 'b) map	
			Base.Map.remove_multi : ('k, 'v list, 'cmp) map -> 'k -> ('k, 'v list, 'cmp) map
			Base.Map.set : ('k, 'v, 'cmp) map -> key:'k -> data:'v -> ('k, 'v, 'cmp) map
			Base.Map.sexp_of_m__t : (module Base.Map.Sexp_of_m with type t = 'k) -> ('v -> Base__.Sexp.t) -> ('k, 'v, 'cmp) map -> Base__.Sexp.t
Map.Make.singleton	CCMap.Make.singleton	BatMap.singleton : 'a -> 'b -> ('a, 'b) map	Base.Map.singleton : ('a, 'cmp) Base.Map.comparator -> 'a -> 'b -> ('a, 'b, 'cmp) map
Map.Make.split	CCMap.Make.split	BatMap.split : 'key -> ('key, 'a) map -> ('key, 'a) map * 'a option * ('key, 'a) map	Base.Map.split : ('k, 'v, 'cmp) map -> 'k -> ('k, 'v, 'cmp) map * ('k * 'v) option * ('k, 'v, 'cmp) map
			Base.Map.subrange : ('k, 'v, 'cmp) map -> lower_bound:'k Base__.Maybe_bound.t -> upper_bound:'k Base__.Maybe_bound.t -> ('k, 'v, 'cmp) map
			Base.Map.symmetric_diff : ('k, 'v, 'cmp) map -> ('k, 'v, 'cmp) map -> data_equal:('v -> 'v -> bool) -> ('k, 'v) Base.Map.Symmetric_diff_element.t Base__.Sequence.t
			Base.Map.to_alist : ?key_order:[`Decreasing | `Increasing] -> ('k, 'v, 'a) map -> ('k * 'v) list
	CCMap.Make.to_iter		
	CCMap.Make.to_list		
Map.Make.to_rev_seq	CCMap.Make.to_rev_seq	BatMap.to_rev_seq : ('key, 'a) map -> ('key * 'a) BatSeq.t	
Map.Make.to_seq	CCMap.Make.to_seq	BatMap.to_seq : ('key, 'a) map -> ('key * 'a) BatSeq.t	
Map.Make.to_seq_from	CCMap.Make.to_seq_from	BatMap.to_seq_from : 'key -> ('key, 'a) map -> ('key * 'a) BatSeq.t	
			Base.Map.to_sequence : ?order:[`Decreasing_key | `Increasing_key] -> ?keys_greater_or_equal_to:'k -> ?keys_less_or_equal_to:'k -> ('k, 'v, 'cmp) map -> ('k * 'v) Base__.Sequence.t
Map.Make.union	CCMap.Make.union	BatMap.union : ('a, 'b) map -> ('a, 'b) map -> ('a, 'b) map	

Stdlib	Containers	Batteries	Base
		BatMap.union_stdlib : ('key -> 'a -> 'a -> 'a option) -> ('key, 'a) map -> ('key, 'a) map -> ('key, 'a) map	
Map.Make.update	CCMap.Make.update	BatMap.update : 'a -> 'a -> 'b -> ('a, 'b) map -> ('a, 'b) map	Base.Map.update : ('k, 'v, 'cmp) map -> 'k -> f:('v option -> 'v) -> ('k, 'v, 'cmp) map
		BatMap.update_stdlib : 'a -> ('b option -> 'b option) -> ('a, 'b) map -> ('a, 'b) map	
			Base.Map.validate : name:('k -> string) -> 'v Base__.Validate.check -> ('k, 'v, 'a) map Base__.Validate.check
			Base.Map.validatei : name:('k -> string) -> ('k * 'v) Base__.Validate.check -> ('k, 'v, 'a) map Base__.Validate.check
	CCMap.Make.values	BatMap.values : ('a, 'b) map -> 'b BatEnum.t	