Module sui::vec_map

A map data structure backed by a vector. The map is guaranteed not to contain duplicate keys, but entries are not sorted by key--entries are included in insertion order. All operations are O(N) in the size of the map--the intention of this data structure is only to provide the convenience of programming against a map API. Large maps should use handwritten parent/child relationships instead. Maps that need sorted iteration rather than insertion order iteration should also be handwritten.

An entry in the map

This key already exists in the map

This key does not exist in the map

Trying to destroy a map that is not empty

Trying to access an element of the map at an invalid index

Trying to pop from a map that is empty

Trying to construct a map from keys and values of different lengths

Create an empty VecMap

Insert the entry key |-> value into self. Aborts if key is already bound in self.

Remove the entry key |-> value from self. Aborts if key is not bound in self.

Pop the most recently inserted entry from the map. Aborts if the map is empty.

Get a mutable reference to the value bound to key in self. Aborts if key is not bound in self.

Get a reference to the value bound to key in self. Aborts if key is not bound in self.

Safely try borrow a value bound to key in self. Return Some(V) if the value exists, None otherwise. Only works for a "copyable" value as references cannot be stored in vector.

Return true if self contains an entry for key, false otherwise

Return the number of entries in self

Return true if self has 0 elements, false otherwise

Destroy an empty map. Aborts if self is not empty

Unpack self into vectors of its keys and values. The output keys and values are stored in insertion order, not sorted by key.

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

Returns a list of keys in the map. Do not assume any particular ordering.

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

Struct

A map data structure backed by a vector. The map is guaranteed not to contain duplicate keys, but entries are not sorted by key--entries are included in insertion order. All operations are O(N) in the size of the map--the intention of this data structure is only to provide the convenience of programming against a map API. Large maps should use handwritten parent/child relationships instead. Maps that need sorted iteration rather than insertion order iteration should also be handwritten.

```bash
```

An entry in the map

```bash
```

This key already exists in the map

```bash
```

This key does not exist in the map

```bash
```

Trying to destroy a map that is not empty

```bash
```

Trying to access an element of the map at an invalid index

```bash
```

Trying to pop from a map that is empty

```bash
```

Trying to construct a map from keys and values of different lengths

```bash
```

Create an empty VecMap

```bash
```

```bash
```

Insert the entry key |-> value into self. Aborts if key is already bound in self.

```bash
```

```bash
```

Remove the entry key |-> value from self. Aborts if key is not bound in self.

```bash
```

```bash
```

Pop the most recently inserted entry from the map. Aborts if the map is empty.

```bash
```

```bash
```

Get a mutable reference to the value bound to key in self. Aborts if key is not bound in self.

```bash
```

```bash
```

Get a reference to the value bound to key in self. Aborts if key is not bound in self.

```bash
```

```bash
```

Safely try borrow a value bound to key in self. Return Some(V) if the value exists, None otherwise. Only works for a "copyable" value as references cannot be stored in vector.

```bash
```

```bash
```

Return true if self contains an entry for key, false otherwise

```bash
```

```bash
```

Return the number of entries in self

```bash
```

```bash
```

Return true if self has 0 elements, false otherwise

```bash
```

```bash
```

Destroy an empty map. Aborts if self is not empty

```bash
```

```bash
```

Unpack self into vectors of its keys and values. The output keys and values are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

```bash
```

```bash
```

Returns a list of keys in the map. Do not assume any particular ordering.

```bash
```

```bash
```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash

```

```bash

```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Struct

An entry in the map

```bash

```

This key already exists in the map

```bash

```

This key does not exist in the map

```bash

```

Trying to destroy a map that is not empty

```bash
```

Trying to access an element of the map at an invalid index

```bash
```

Trying to pop from a map that is empty

```bash
```

Trying to construct a map from keys and values of different lengths

```bash
```

Create an empty VecMap

```bash
```

```bash
```

Insert the entry key |-> value into self. Aborts if key is already bound in self.

```bash
```

```bash
```

Remove the entry key |-> value from self. Aborts if key is not bound in self.

```bash
```

```bash
```

Pop the most recently inserted entry from the map. Aborts if the map is empty.

```bash
```

```bash
```

Get a mutable reference to the value bound to key in self. Aborts if key is not bound in self.

```bash
```

```bash
```

Get a reference to the value bound to key in self. Aborts if key is not bound in self.

```bash
```

```bash
```

Safely try borrow a value bound to key in self. Return Some(V) if the value exists, None otherwise. Only works for a "copyable" value as references cannot be stored in vector.

```bash
```

```bash
```

Return true if self contains an entry for key, false otherwise

```bash
```

```bash
```

Return the number of entries in self

```bash
```

```bash
```

Return true if self has 0 elements, false otherwise

```bash
```

```bash
```

Destroy an empty map. Aborts if self is not empty

```bash
```

```bash
```

Unpack self into vectors of its keys and values. The output keys and values are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

```bash
```

```bash
```

Returns a list of keys in the map. Do not assume any particular ordering.

```bash
```

```bash
```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash

```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Constants

This key already exists in the map

```bash

```

This key does not exist in the map

```bash

```

Trying to destroy a map that is not empty

```bash

```

Trying to access an element of the map at an invalid index

```bash

```

Trying to pop from a map that is empty

```bash

```

Trying to construct a map from keys and values of different lengths

```bash

```

Create an empty VecMap

```bash

```

```bash

```

Insert the entry key |-> value into self. Aborts if key is already bound in self.

```bash

```

```bash
```

Remove the entry key |-> value from self. Aborts if key is not bound in self.

```bash
```

```bash
```

Pop the most recently inserted entry from the map. Aborts if the map is empty.

```bash
```

```bash
```

Get a mutable reference to the value bound to key in self. Aborts if key is not bound in self.

```bash
```

```bash
```

Get a reference to the value bound to key in self. Aborts if key is not bound in self.

```bash
```

```bash
```

Safely try borrow a value bound to key in self. Return Some(V) if the value exists, None otherwise. Only works for a "copyable" value as references cannot be stored in vector.

```bash
```

```bash
```

Return true if self contains an entry for key, false otherwise

```bash
```

```bash
```

Return the number of entries in self

```bash
```

```bash
```

Return true if self has 0 elements, false otherwise

```bash
```

```bash
```

Destroy an empty map. Aborts if self is not empty

```bash
```

```bash
```

Unpack self into vectors of its keys and values. The output keys and values are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

```bash
```

```bash
```

Returns a list of keys in the map. Do not assume any particular ordering.

```bash
```

```bash
```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Function

Create an empty VecMap

```bash
```

```bash
```

Insert the entry key |-> value into self. Aborts if key is already bound in self.

```bash
```

```bash
```

Remove the entry key |-> value from self. Aborts if key is not bound in self.

```bash
```

```bash
```

Pop the most recently inserted entry from the map. Aborts if the map is empty.

```bash
```

```bash
```

Get a mutable reference to the value bound to key in self. Aborts if key is not bound in self.

```bash
```

```bash
```

Get a reference to the value bound to key in self. Aborts if key is not bound in self.

```bash
```

```bash
```

Safely try borrow a value bound to key in self. Return Some(V) if the value exists, None otherwise. Only works for a "copyable" value as references cannot be stored in vector.

```bash
```

```bash
```

Return true if self contains an entry for key, false otherwise

```bash
```

```bash
```

Return the number of entries in self

```bash
```

```bash
```

Return true if self has 0 elements, false otherwise

```bash
```

```bash
```

Destroy an empty map. Aborts if self is not empty

```bash
```

```bash
```

Unpack self into vectors of its keys and values. The output keys and values are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

```bash
```

```bash
```

Returns a list of keys in the map. Do not assume any particular ordering.

```bash
```

```bash
```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Function

Insert the entry key |-> value into self. Aborts if key is already bound in self.

```bash
```

```bash
```

Remove the entry key |-> value from self. Aborts if key is not bound in self.

```bash
```

```bash
```

Pop the most recently inserted entry from the map. Aborts if the map is empty.

```bash
```

```bash
```

Get a mutable reference to the value bound to key in self. Aborts if key is not bound in self.

```bash
```

```bash
```

Get a reference to the value bound to key in self. Aborts if key is not bound in self.

```bash
```

```bash
```

Safely try borrow a value bound to key in self. Return Some(V) if the value exists, None otherwise. Only works for a "copyable" value as references cannot be stored in vector.

```bash
```

```bash
```

Return true if self contains an entry for key, false otherwise

```bash
```

```bash
```

Return the number of entries in self

```bash
```

```bash
```

Return true if self has 0 elements, false otherwise

```bash
```

```bash
```

Destroy an empty map. Aborts if self is not empty

```bash
```

```bash
```

Unpack self into vectors of its keys and values. The output keys and values are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

```bash
```

```bash
```

Returns a list of keys in the map. Do not assume any particular ordering.

```bash
```

```bash
```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash

```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Function

Remove the entry key |-> value from self. Aborts if key is not bound in self.

```bash

```

```bash

```

Pop the most recently inserted entry from the map. Aborts if the map is empty.

```bash

```

```bash

```

Get a mutable reference to the value bound to key in self. Aborts if key is not bound in self.

```bash

```

```bash

```

Get a reference to the value bound to key in self. Aborts if key is not bound in self.

```bash

```

```bash
```

Safely try borrow a value bound to key in self. Return Some(V) if the value exists, None otherwise. Only works for a "copyable" value as references cannot be stored in vector.

```bash

```

```bash

```

Return true if self contains an entry for key, false otherwise

```bash

```

```bash

```

Return the number of entries in self

```bash

```

```bash

```

Return true if self has 0 elements, false otherwise

```bash

```

```bash

```

Destroy an empty map. Aborts if self is not empty

```bash

```

```bash

```

Unpack self into vectors of its keys and values. The output keys and values are stored in insertion order, not sorted by key.

```bash

```

```bash

```

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

```bash
```

```bash
```

Returns a list of keys in the map. Do not assume any particular ordering.

```bash
```

```bash
```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Function

Pop the most recently inserted entry from the map. Aborts if the map is empty.

```bash

```

```bash

```

Get a mutable reference to the value bound to key in self. Aborts if key is not bound in self.

```bash

```

```bash

```

Get a reference to the value bound to key in self. Aborts if key is not bound in self.

```bash

```

```bash

```

Safely try borrow a value bound to key in self. Return Some(V) if the value exists, None otherwise. Only works for a "copyable" value as references cannot be stored in vector.

```bash

```

```bash

```

Return true if self contains an entry for key, false otherwise

```bash

```

```bash

```

Return the number of entries in self

```bash

```

```bash

```

Return true if self has 0 elements, false otherwise

```bash
```

```bash
```

Destroy an empty map. Aborts if self is not empty

```bash
```

```bash
```

Unpack self into vectors of its keys and values. The output keys and values are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

```bash
```

```bash
```

Returns a list of keys in the map. Do not assume any particular ordering.

```bash
```

```bash
```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Function

Get a mutable reference to the value bound to key in self. Aborts if key is not bound in self.

```bash

```

```bash

```

Get a reference to the value bound to key in self. Aborts if key is not bound in self.

```bash

```

```bash

```

Safely try borrow a value bound to key in self. Return Some(V) if the value exists, None otherwise. Only works for a "copyable" value as references cannot be stored in vector.

```bash

```

```bash

```

Return true if self contains an entry for key, false otherwise

```bash
```

```bash
```

Return the number of entries in self

```bash
```

```bash
```

Return true if self has 0 elements, false otherwise

```bash
```

```bash
```

Destroy an empty map. Aborts if self is not empty

```bash
```

```bash
```

Unpack self into vectors of its keys and values. The output keys and values are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

```bash
```

```bash
```

Returns a list of keys in the map. Do not assume any particular ordering.

```bash
```

```bash
```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Function

Get a reference to the value bound to key in self. Aborts if key is not bound in self.

```bash
```

```bash
```

Safely try borrow a value bound to key in self. Return Some(V) if the value exists, None otherwise. Only works for a "copyable" value as references cannot be stored in vector.

```bash

```

```bash

```

Return true if self contains an entry for key, false otherwise

```bash

```

```bash

```

Return the number of entries in self

```bash

```

```bash

```

Return true if self has 0 elements, false otherwise

```bash

```

```bash

```

Destroy an empty map. Aborts if self is not empty

```bash

```

```bash

```

Unpack self into vectors of its keys and values. The output keys and values are stored in insertion order, not sorted by key.

```bash

```

```bash

```

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

```bash
```

```bash
```

Returns a list of keys in the map. Do not assume any particular ordering.

```bash
```

```bash
```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Function

Safely try borrow a value bound to key in self. Return Some(V) if the value exists, None otherwise. Only works for a "copyable" value as references cannot be stored in vector.

```bash

```

```bash

```

Return true if self contains an entry for key, false otherwise

```bash

```

```bash

```

Return the number of entries in self

```bash

```

```bash

```

Return true if self has 0 elements, false otherwise

```bash

```

```bash

```

Destroy an empty map. Aborts if self is not empty

```bash

```

```bash

```

Unpack self into vectors of its keys and values. The output keys and values are stored in insertion order, not sorted by key.

```bash

```

```bash

```

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

```bash

```

```bash

```

Returns a list of keys in the map. Do not assume any particular ordering.

```bash

```

```bash

```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash

```

```bash

```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash

```

```bash

```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash

```

Function

Return true if self contains an entry for key, false otherwise

```bash

```

```bash

```

Return the number of entries in self

```bash

```

```bash

```

Return true if self has 0 elements, false otherwise

```bash

```

```bash

```

Destroy an empty map. Aborts if self is not empty

```bash

```

```bash

```

Unpack self into vectors of its keys and values. The output keys and values are stored in insertion order, not sorted by key.

```bash

```

```bash

```

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

```bash

```

```bash
```

Returns a list of keys in the map. Do not assume any particular ordering.

```bash
```

```bash
```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Function

Return the number of entries in self

```bash
```

```bash
```

Return true if self has 0 elements, false otherwise

```bash
```

```bash
```

Destroy an empty map. Aborts if self is not empty

```bash
```

```bash
```

Unpack self into vectors of its keys and values. The output keys and values are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

```bash
```

```bash
```

Returns a list of keys in the map. Do not assume any particular ordering.

```bash
```

```bash
```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Function

Return true if self has 0 elements, false otherwise

```bash
```

```bash
```

Destroy an empty map. Aborts if self is not empty

```bash
```

```bash
```

Unpack self into vectors of its keys and values. The output keys and values are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

```bash
```

```bash
```

Returns a list of keys in the map. Do not assume any particular ordering.

```bash
```

```bash
```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Function

Destroy an empty map. Aborts if self is not empty

```bash
```

```bash
```

Unpack self into vectors of its keys and values. The output keys and values are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

```bash
```

```bash
```

Returns a list of keys in the map. Do not assume any particular ordering.

```bash
```

```bash
```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash

```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash

```

```bash

```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Function

Unpack self into vectors of its keys and values. The output keys and values are stored in insertion order, not sorted by key.

```bash

```

```bash

```

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

```bash
```

```bash

```

Returns a list of keys in the map. Do not assume any particular ordering.

```bash

```

```bash

```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash

```

```bash

```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash

```

```bash

```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Function

Construct a new VecMap from two vectors, one for keys and one for values. The key value pairs are associated via their indices in the vectors, e.g. the key at index i in keys is associated with the value at index i in values. The key value pairs are stored in insertion order (the original vectors ordering) and are not sorted.

```bash

```

```bash

```

Returns a list of keys in the map. Do not assume any particular ordering.

```bash

```

```bash

```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash

```

```bash

```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash

```

```bash

```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash

```

```bash

```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Function

Returns a list of keys in the map. Do not assume any particular ordering.

```bash
```

```bash
```

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Function

Find the index of key in self. Return None if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Function

Find the index of key in self. Aborts if key is not in self. Note that map entries are stored in insertion order, not sorted by key.

```bash
```

```bash
```

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Function

Return a reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Function

Return a mutable reference to the idxth entry of self. This gives direct access into the backing array of the map--use with caution. Note that map entries are stored in insertion order, not sorted by key. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```

Function

Remove the entry at index idx from self. Aborts if idx is greater than or equal to size (self)

```bash
```

```bash
```