2.13. string.ak

use aiken/builtin.{

append\_bytearray, append\_string, decode\_utf8, encode\_utf8, length\_of\_bytearray,

}

use aiken/cbor

/// Combine two `String` together.

///

/// ```aiken

/// string.concat(left: @"Hello", right: @", World!") == @"Hello, World!"

/// ```

pub fn concat(left: String, right: String) -> String {

append\_string(left, right)

}

test concat\_1() {

concat(@"", @"") == @""

}

test concat\_2() {

concat(@"", @"foo") == concat(@"foo", @"")

}

test concat\_3() {

concat(left: @"Hello", right: @", World!") == @"Hello, World!"

}

/// Convert a `ByteArray` into a `String`

///

/// <br/>⚠️<pre>WARNING</pre> | This functions fails if the underlying `ByteArray` isn't UTF-8-encoded. <br/>In particular, you cannot convert arbitrary hash digests using this function. <br/>For converting arbitrary `ByteArray`s, use [bytearray.to\_hex](/stdlib/aiken/bytearray.html#to\_hex).

/// --- | ---

///

/// ```aiken

/// string.from\_bytearray("foo") == @"foo"

///

/// string.from\_bytearray(#"666f6f") == @"foo"

///

/// string.from\_bytearray(some\_hash) -> fail

/// ```

pub fn from\_bytearray(bytes: ByteArray) -> String {

decode\_utf8(bytes)

}

test from\_bytearray\_1() {

from\_bytearray(#[]) == @""

}

test from\_bytearray\_2() {

from\_bytearray(#[65, 66, 67]) == @"ABC"

}

test from\_bytearray\_3() {

from\_bytearray("ABC") == @"ABC"

}

/// Convert an `Int` to its `String` representation.

///

/// ```aiken

/// string.from\_int(42) == @"42"

/// ```

pub fn from\_int(n: Int) -> String {

cbor.diagnostic(n)

}

test from\_int\_1() {

from\_int(0) == @"0"

}

test from\_int\_2() {

from\_int(5) == @"5"

}

test from\_int\_3() {

from\_int(42) == @"42"

}

test from\_int\_4() {

from\_int(200) == @"200"

}

/// Join a list of strings, separated by a given \_delimiter\_.

///

/// ```aiken

/// string.join([], @"+") == @""

/// string.join([@"a", @"b", @"c"], @",") == @"a,b,c"

/// ```

pub fn join(list: List<String>, delimiter: String) -> String {

do\_join(list, encode\_utf8(delimiter), #"")

|> decode\_utf8

}

fn do\_join(xs, delimiter, bytes) {

when xs is {

[] -> bytes

[x, ..rest] ->

do\_join(

rest,

delimiter,

if length\_of\_bytearray(bytes) == 0 {

encode\_utf8(x)

} else {

append\_bytearray(bytes, append\_bytearray(delimiter, encode\_utf8(x)))

},

)

}

}

test join\_1() {

join([], @",") == @""

}

test join\_2() {

join([@"a", @"b", @"c"], @",") == @"a,b,c"

}

/// Convert a `String` into a `ByteArray`

///

/// ```aiken

/// string.to\_bytearray(@"foo") == "foo"

/// ```

pub fn to\_bytearray(self: String) -> ByteArray {

encode\_utf8(self)

}

test to\_bytearray\_1() {

to\_bytearray(@"") == ""

}

test to\_bytearray\_2() {

to\_bytearray(@"ABC") == #[65, 66, 67]

}

test to\_bytearray\_3() {

to\_bytearray(@"ABC") == "ABC"

}