2.7. int.ak

use aiken/bytearray

use aiken/math

use aiken/option

/// Compare two integers.

///

/// ```aiken

/// int.compare(14, 42) == Less

/// int.compare(14, 14) == Equal

/// int.compare(42, 14) == Greater

/// ```

pub fn compare(left: Int, right: Int) -> Ordering {

if left < right {

Less

} else if left > right {

Greater

} else {

Equal

}

}

/// Parse an integer from a utf-8 encoded 'ByteArray', when possible.

///

/// ```aiken

/// int.from\_utf8("14") == Some(14)

/// int.from\_utf8("-42") == Some(-42)

/// int.from\_utf8("007") == Some(7)

/// int.from\_utf8("foo") == None

/// int.from\_utf8("1.0") == None

/// int.from\_utf8("1-2") == None

/// ```

pub fn from\_utf8(bytes: ByteArray) -> Option<Int> {

bytes

|> bytearray.foldr(

Some((0, 0)),

fn(byte, st) {

when st is {

None -> None

Some((n, e)) ->

if byte < 48 || byte > 57 {

if byte == 45 {

Some((-n, 0))

} else {

None

}

} else if n < 0 {

None

} else {

let digit = byte - 48

Some((n + digit \* math.pow(10, e), e + 1))

}

}

},

)

|> option.map(fn(tuple) { tuple.1st })

}

test from\_utf8\_1() {

from\_utf8("0017") == Some(17)

}

test from\_utf8\_2() {

from\_utf8("42") == Some(42)

}

test from\_utf8\_3() {

from\_utf8("1337") == Some(1337)

}

test from\_utf8\_4() {

from\_utf8("-14") == Some(-14)

}

test from\_utf8\_5() {

from\_utf8("foo") == None

}

test from\_utf8\_6() {

from\_utf8("1-2") == None

}