2.6. hash.ak

//// This module defines `Hash`, a self-documenting type-alias with a

//// phantom-type for readability.

////

//// On-chain, any hash digest value is represented as a plain 'ByteArray'.

//// Though in practice, hashes come from different sources and have

//// different semantics.

////

//// Hence, while this type-alias doesn't provide any strong type-guarantees,

//// it helps writing functions signatures with more meaningful types than mere

//// 'ByteArray'.

////

//// Compare for example:

////

//// ```aiken

//// pub type Credential {

//// VerificationKeyCredential(ByteArray)

//// ScriptCredential(ByteArray)

//// }

//// ```

////

//// with

////

//// ```aiken

//// pub type Credential {

//// VerificationKeyCredential(Hash<Blake2b\_224, VerificationKey>)

//// ScriptCredential(Hash<Blake2b\_224, Script>)

//// }

//// ```

////

//// Both are strictly equivalent, but the second reads much better.

use aiken/builtin

/// A `Hash` is nothing more than a `ByteArray`, but it carries extra

/// information for readability.

pub type Hash<alg, a> =

ByteArray

/// A blake2b-224 hash algorithm.

///

/// Typically used for:

///

/// - [`Credential`](../aiken/transaction/credential.html#Credential)

/// - [`PolicyId`](../aiken/transaction/value.html#PolicyId)

///

/// Note: there's no function to calculate blake2b-224 hash digests on-chain.

pub opaque type Blake2b\_224 {

Blake2b\_224

}

/// A blake2b-256 hash algorithm.

///

/// Typically used for:

///

/// - [`TransactionId`](../aiken/transaction.html#TransactionId)

pub opaque type Blake2b\_256 {

Blake2b\_256

}

/// Compute the blake2b-256 hash digest of some data.

pub fn blake2b\_256(bytes: ByteArray) -> Hash<Blake2b\_256, a> {

builtin.blake2b\_256(bytes)

}

/// A SHA2-256 hash algorithm.

pub opaque type Sha2\_256 {

Sha2\_256

}

/// Compute the sha2-256 hash digest of some data.

pub fn sha2\_256(bytes: ByteArray) -> Hash<Sha2\_256, a> {

builtin.sha2\_256(bytes)

}

/// A SHA3-256 hash algorithm.

pub opaque type Sha3\_256 {

Sha3\_256

}

/// Compute the sha3-256 hash digest of some data.

pub fn sha3\_256(bytes: ByteArray) -> Hash<Sha3\_256, a> {

builtin.sha3\_256(bytes)

}