**Control Token Testing (These are only public, not helper functions)**

**Testing Environment: Solidity Realtime Compiler**

1. **ControlToken(bytes32[100] theControllerHashes, uint[100] theTokensOwned)**

*Description:* This is the constructor. We input the controller tokens and how many tokens they own respectively. NOTE: For testing purposes, I made the size 4 instead of 100.

*Input:*

bytes32[100] theControllerHashes = [“C1”, “C2”, “C3”, “”];

uint[100] theTokensOwned = [10,10,10,0];

*Output:* The constructor itself does not return a value. However, investigation into the structs controllerHashes and tokensOwned showed they were holding the input correctly. “C1” was encoded, perhaps because it was not written with the full 32 bytes. Of course, this is not a problem because with a hash, we know it works.

*Passed Test?:* YES

1. **function changeTokenController(bytes32 originalControllerHash, bytes32 newControllerHash, uint amount) returns (bool success)**

**//look into pipes**

*Description*: This allows a controller to give tokens they own to another controller (who must exist in the list). In this function, we must have controller “C1” give controller “C2” 5 tokens. The function makes sure the controller can only give as many tokens as they own.

*Input*:

bytes32 originalControllerHash = “C1”;

bytes32 newControllerHash = “C2”;

uint amount = 5;

*Output*: bool success = true; Inspection of tokens owned shows that indeed, at index 0 we have 5 tokens and at index 1 we have 15 tokens.

*Passed Test?:* YES

1. **function delegate(bytes32 controllerHash, bytes32 delegateeHash, uint amount) returns (bool success)**

*Description:* This function allows a controller to delegate their tokens to a delegate. If they do not exist, it adds them to the list. They are only allowed to delegate (total amount of tokens owned) - (total amount of tokens delegated).

*Input:* We let controller “C1”, who has five tokens, delegate four tokens to a new delegate who this function will add to the list—“D1”.

bytes32 controllerHash = “C1”;

bytes32 delegateeHash = “D1”;

uint amount = 4;

*Output:*

bool success = true;

*Passed Test?: YES* (see 4.)

1. **function myAmount(bytes32 delegateeHash) returns (uint amount)**

*Description:* This function allows a delegate to know how many tokens they have currently been delegated.

*Input:* We have the delegate “D1” call the function to see how many tokens that they own.

bytes32 delegateeHash = “D1”;

*Output:*

uint amount = 4;

*Passed Test?: YES*

1. **function revokeDelegation(bytes32 controllerHash, bytes32 delegateeHash, uint amount) returns (bool success)**

*Description:* This function allows a controller to revoke their delegation to a delegate before the delegate has a chance to use it. Since “C1” has delegated 4 tokens to “D1”, we call this function with 2 tokens (not all four). If “C1” calls any number greater than how many they have delegated to “D1”, the function is written to revoke only how many “D1” has been delegated by “C1”.

*Input*:

bytes32 controllerHash = “C1”;

bytes32 delegateeHash = “D1”;

uint amount = 2;

*Output:*

*bool success = true;*

*Passed Test?:* *YES* (calling the function myAmount(“D1”) gives 2 and calling the function amountDelegated(“C1”) gives 2)

1. **function spendMyTokens(bytes32 delegateeHash, uint amount)**

*Description:* This function allows a delegate to spend their tokens. We have delegate “D1” spend 1 of their 2 tokens.

*Input:*

bytes32 delegateeHash = “D1”;

uint amount = 1;

*Output:*

Function returns no output.

*Passed Test?:YES (*calling the function myAmount(“D1”) gives 1 and calling the function amountDelegated(“C1”) gives 2)

1. **function amountDelegated(bytes32 controllerHash) returns (uint val)**

*Description:* This function allows a controller to see how many tokens that they have

*Input:*

1. “C1”
2. “C2”

*Output:*

1. 1
2. 0

*Passed Test?: YES*

Note, at this time (8/23), the solidity realtime compiler is down. So I am doing the testing on Ether Camp. I use the bytes32 conversion (which ether camp doesn’t conver)

“C1” = 0x4331000000000000000000000000000000000000000000000000000000000000

“C2” = 0x4332000000000000000000000000000000000000000000000000000000000000

“C3” = 0x4333000000000000000000000000000000000000000000000000000000000000

“D1” = 0x4431000000000000000000000000000000000000000000000000000000000000

“D2” = 0x4432000000000000000000000000000000000000000000000000000000000000

“D3” = 0x4433000000000000000000000000000000000000000000000000000000000000

1. **function removeController(bytes32 controllerHash) returns (bool success)**

*Description*: Removes a controller. Should be called only from coid after validation that the one calling the function *is* *an owner*.

*Input*: 1. “C1” 2. “C2” 3. “C3”

*Output:*

1.,2.,3. bool success = true;

*Success?: YES. Investigation of the arrays relations, tokensOwned and controllerHashes show that the function works properly.*

1. **function addController(bytes32 controllerHash) returns (bool success)**

*Description*: Adds a controller. Should be called only from coid after validation that the one calling the function *is* *an owner*.

*Input*: “C4”

*Output:*

bool success = true;

*Success?: YES. Investigation of the arrays relations, tokensOwned and controllerHashes show that the function works properly.*