**Properties:**

1. Alice’s randomly generated message is an array of 1’s and 0’s
2. Alice’s randomly generated message is equal to the length specified, in the parameter for the function
3. The length of the encoded message (array of encoded qubits) is the same length as the array of bits/basis to encode
4. A message that has been decoded is the same length as the encoded message array/original random bit array/basis
5. A message that has been decoded (measured) is an array of 1’s and 0’s
6. An encoded message that is decoded with the same basis array that it was encoded by, should return an array of the exact original bits that were used to encode the array.
7. Two encoded messages, decoded with their original basis are return equal outputs
8. The output key length with never be larger than length of original bits
9. The two generated keys should be equal
10. The Keys that have been generated at the end is an array of 1’s and 0’s

Properties (New):

1.

Precondition:

Random binary string to encode: M

Random binary string for bases: B

Length(M) = Length(B)

Operation:

encodeMessage(M, B): encodedCircuit

Output:

assertEqual(Length(measureZ(encodedCircuit)), Length(M) )

(Encoded message length should be equal to encoded binary string length)

2.

Precondition:

Random binary string to encode: M

Random binary string for bases: BAlice

Random binary string for bases: BBob

Length(M) = Length(BAlice)

Length(M) = Length(BBob)

Operation:

encodeMessage(M, BAlice): encodedCircuit

measureMessage(encodedCircuit, BBob):

Output:

assertEqual(Length(measureMessage(encodedCircuit, BBob)), Length(encodedCircuit))

(encodedCircuit needs to be the same length as measured circuit with Bob’s bases)

3.

Precondition:

Random binary string to encode: M

Random binary string for bases: BAlice

Random binary string for bases: BBob

Length(M) = Length(BAlice)

Length(M) = Length(BBob)

Operation:

encodeMessage(M, BAlice): encodedCircuit

measureMessage(encodedCircuit, BBob): measuredArray

Output:

assertTrue(isBinaryArray(measuredArray))

(Measured array should be an array that only contains 1’s and 0’s)

4.

Precondition:

Random binary string to encode: M

Random binary string for bases: BAlice

Length(M) = Length(BAlice)

Operation:

encodeMessage(M, BAlice): encodedCircuit

measureMessage(encodedCircuit, BAlice): measuredArray

Output:

assertEqual(measuredArray, M)

(if an array is measured in the same basis it was encoded by it should output the same the originally encoded message)

5.

Precondition:

Random binary string to encode: M

Random binary string for bases: BAlice

Random binary string for bases: BBob

Length(M) >= 100

Length(M) = Length(BAlice)

Length(M) = Length(BBob)

BAlice != BBob

Operation:

encodeMessage(M, BAlice): encodedCircuitAlice

encodeMessage(M, BBob): encodedCircuitBob

Output:

assertTrue(measureZ(encodedCircuitAlice) != measureZ(encodedCircuitBob))

assertTrue(measureX(encodedCircuitAlice) != measureX(encodedCircuitBob))

(the same message encoded with different basis, should output different circuits (hence output different things when measured in Z/X basis))

6.

Precondition: ///

Random binary string to encode: M

Random binary string for bases: BAlice

Random binary string for bases: BBob

Length(M) = Length(BAlice)

Length(M) = Length(BBob)

BAlice != BBob

Operation:

encodeMessage(M, BAlice): encodedCircuitAlice

encodeMessage(M, BBob): encodedCircuitBob

Output:

assertTrue(measureMessage(encodedCircuitAlice, BAlice) != measureMessage (encodedCircuitBob, BBob))

assertEqual(measureMessage(encodedCircuitAlice, BAlice), M)

(The same message encoded with different basis, and measured with the correct basis, should return the original/same message for both)

7.

Precondition: /////

Random binary string to encode: M

Random binary string for bases: BAlice

Random binary string for bases: BBob

Length(M) >= 100

Length(M) = Length(BAlice)

Length(M) = Length(BBob)

BAlice != BBob

Operation:

encodeMessage(M, BAlice): encodedCircuitAlice

Output:

assertTrue(measureMessage(encodedCircuitAlice, BBob) != M)

(Measuring with the wrong basis will not output the original message)

8.

Precondition:

Random binary string to encode: M

Random binary string for bases: BAlice

Random binary string for bases: BBob

Length(M) = Length(BAlice)

Length(M) = Length(BBob)

BAlice != BBob

Operation:

encodeMessage(M, BAlice): encodedCircuitAlice

measureMessage(encodedCircuitAlice, BBob): measuredMessageBob

removeGarbage(BAlice, BBob, M): AliceKey

removeGarbage(BAlice, BBob, measuredMessageBob): BobKey

Generate sample length for verification: S

S < Length(AliceKey)

Generate random binary string of length S: binS

sampleBits(BobKey,binS): bobSample

sampleBits(AliceKey, binS) aliceSample

Output:

assertEqual(AliceKey, BobKey)

assertEqual(bobSample, AliceSample)

removing parts where different basis was used to measured should result in equally decoded keys, samples taken from the keys should also be equal