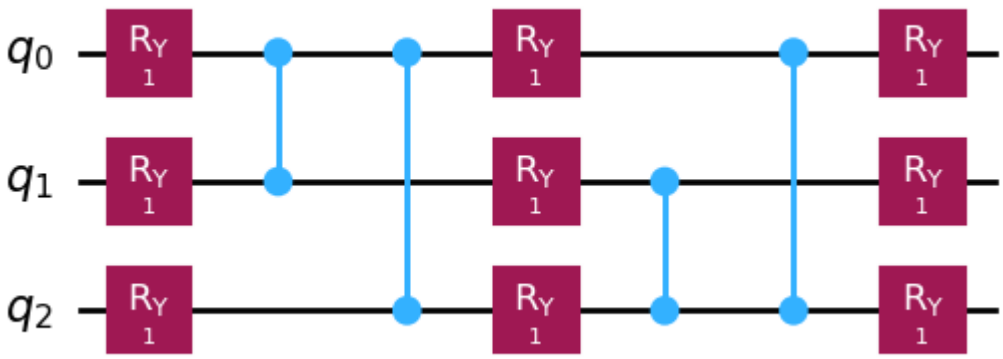
In [1]:
```
import qiskit
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit import transpile, assemble
import math
import random
import numpy as np
from scipy.optimize import minimize
from qiskit_aer import AerSimulator
```

Fixed Hardware Ansatz

In [2]:
```
def apply_fixed_ansatz(qubits, parameters):

    for iz in range (0, len(qubits)):
        circ.ry(parameters[0][iz], qubits[iz])

    circ.cz(qubits[0], qubits[1])
    circ.cz(qubits[2], qubits[0])

    for iz in range (0, len(qubits)):
        circ.ry(parameters[1][iz], qubits[iz])

    circ.cz(qubits[1], qubits[2])
    circ.cz(qubits[2], qubits[0])

    for iz in range (0, len(qubits)):
        circ.ry(parameters[2][iz], qubits[iz])

circ = QuantumCircuit(3)
apply_fixed_ansatz([0, 1, 2], [[1, 1, 1], [1, 1, 1], [1, 1, 1]])
circ.draw('mpl')
```

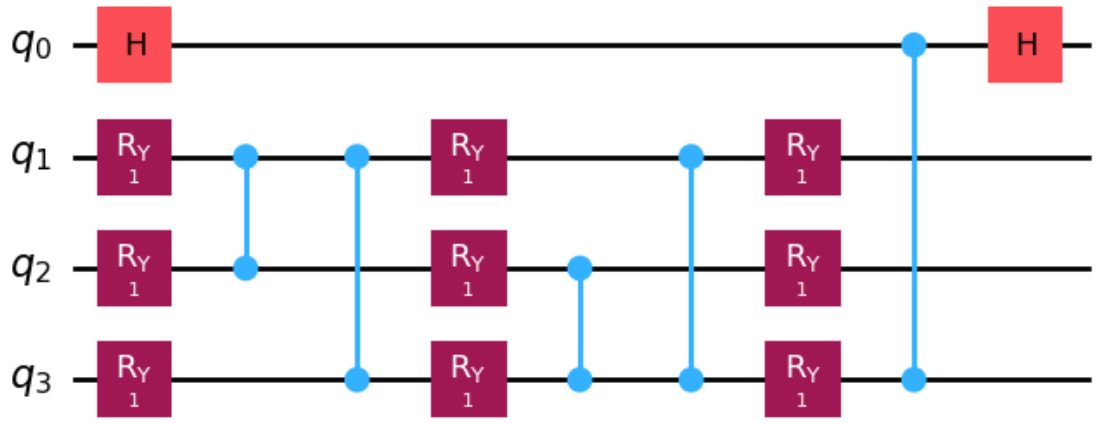
Out[2]:



Creates the Hadamard test

In [3]:
```
def had_test(gate_type, qubits, auxiliary_index, parameters):

    circ.h(auxiliary_index)

    apply_fixed_ansatz(qubits, parameters)

    for ie in range (0, len(gate_type[0])):
        if (gate_type[0][ie] == 1):
            circ.cz(auxiliary_index, qubits[ie])

    for ie in range (0, len(gate_type[1])):
        if (gate_type[1][ie] == 1):
            circ.cz(auxiliary_index, qubits[ie])

    circ.h(auxiliary_index)

circ = QuantumCircuit(4)
had_test([[0, 0, 0], [0, 0, 1]], [1, 2, 3], 0, [[1, 1, 1], [1, 1, 1], [1, 1,
circ.draw('mpl')
```

Out[3]:

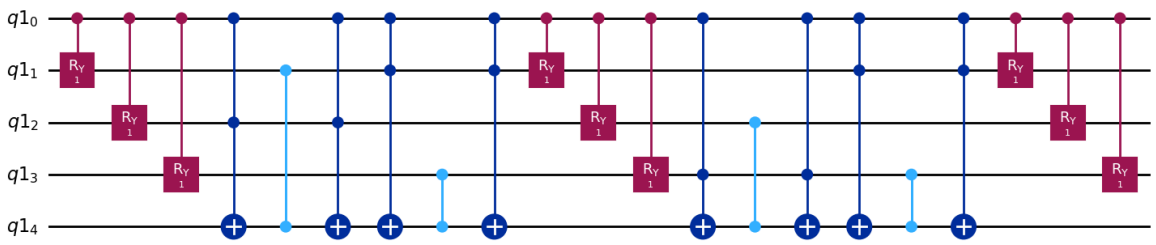


Creates controlled anstaz for calculating |<b|psi>|^2 with a Hadamard test

In [4]:
```
def control_fixed_ansatz(qubits, parameters, auxiliary, reg):

    for i in range (0, len(qubits)):
        circ.cry(parameters[0][i], qiskit.circuit.Qubit(reg, auxiliary), qis

    circ.ccx(auxiliary, qubits[1], 4)
    circ.cz(qubits[0], 4)
    circ.ccx(auxiliary, qubits[1], 4)

    circ.ccx(auxiliary, qubits[0], 4)
    circ.cz(qubits[2], 4)
    circ.ccx(auxiliary, qubits[0], 4)
```

```
    for i in range (0, len(qubits)):
        circ.cry(parameters[1][i], qiskit.circuit.Qubit(reg, auxiliary), qis

    circ.ccx(auxiliary, qubits[2], 4)
    circ.cz(qubits[1], 4)
    circ.ccx(auxiliary, qubits[2], 4)

    circ.ccx(auxiliary, qubits[0], 4)
    circ.cz(qubits[2], 4)
    circ.ccx(auxiliary, qubits[0], 4)

    for i in range (0, len(qubits)):
        circ.cry(parameters[2][i], qiskit.circuit.Qubit(reg, auxiliary), qis

q_reg = QuantumRegister(5)
circ = QuantumCircuit(q_reg)
control_fixed_ansatz([1, 2, 3], [[1, 1, 1], [1, 1, 1], [1, 1, 1]], 0, q_reg)
circ.draw('mpl')
```

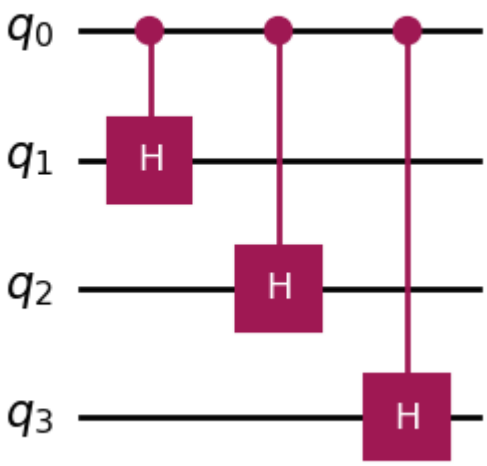
Out[4]:



In [5]:

```
def control_b(auxiliary, qubits):

    for ia in qubits:
        circ.ch(auxiliary, ia)

circ = QuantumCircuit(4)
control_b(0, [1, 2, 3])
circ.draw('mpl')
```

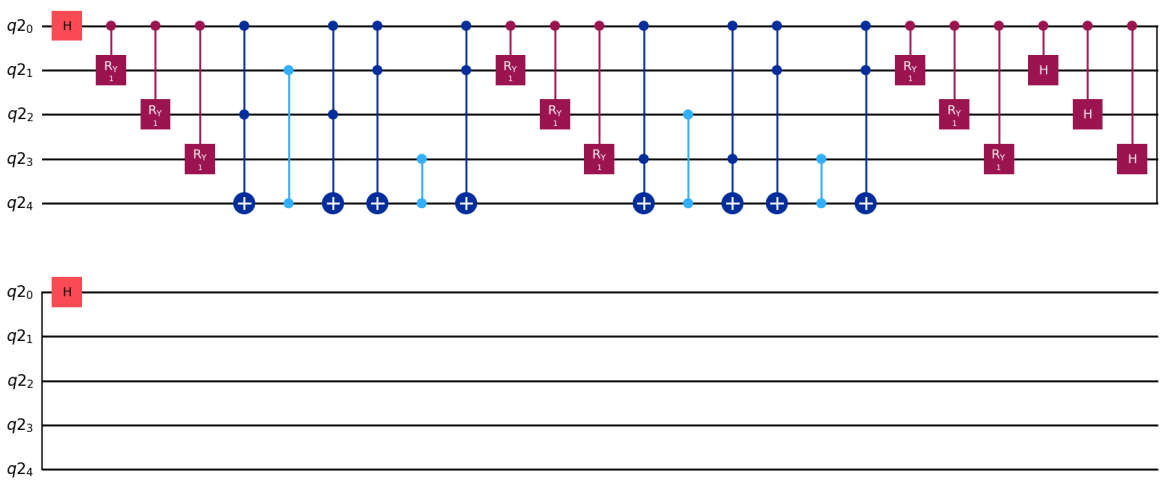
Out[5]:



Create the controlled Hadamard test, for calculating <psi|psi>

In [6]:
```
def special_had_test(gate_type, qubits, auxiliary_index, parameters, reg):

    circ.h(auxiliary_index)

    control_fixed_ansatz(qubits, parameters, auxiliary_index, reg)

    for ty in range (0, len(gate_type)):
        if (gate_type[ty] == 1):
            circ.cz(auxiliary_index, qubits[ty])


    control_b(auxiliary_index, qubits)

    circ.h(auxiliary_index)

q_reg = QuantumRegister(5)
circ = QuantumCircuit(q_reg)
special_had_test([[0, 0, 0], [0, 0, 1]], [1, 2, 3], 0, [[1, 1, 1], [1, 1, 1]
circ.draw('mpl')
```

Out[6]:



Implements the entire cost function on the quantum circuit

In [31]:

```
def calculate_cost_function(parameters):

    global opt

    overall_sum_1 = 0

    parameters = [parameters[0:3], parameters[3:6], parameters[6:9]]

    for i in range(0, len(gate_set)):
        for j in range(0, len(gate_set)):

            global circ

            qctl = QuantumRegister(5)
            qc = ClassicalRegister(5)
            circ = QuantumCircuit(qctl, qc)

            backend = AerSimulator(method='statevector')

            multiply = coefficient_set[i]*coefficient_set[j]

            had_test([gate_set[i], gate_set[j]], [1, 2, 3], 0, parameters)

            circ.save_statevector()
            t_circ = transpile(circ, backend)
            job = backend.run(t_circ, shots=10000)

            result = job.result()
            outputstate = np.real(result.get_statevector(circ, decimals=100)
            o = outputstate

            m_sum = 0
            for l in range (0, len(o)):
                if (l%2 == 1):
```

```
                    n = o[l]**2
                    m_sum+=n

            overall_sum_1+=multiply*(1-(2*m_sum))

    overall_sum_2 = 0

    for i in range(0, len(gate_set)):
        for j in range(0, len(gate_set)):

            multiply = coefficient_set[i]*coefficient_set[j]
            mult = 1

            for extra in range(0, 2):

                qctl = QuantumRegister(5)
                qc = ClassicalRegister(5)
                circ = QuantumCircuit(qctl, qc)

                backend = AerSimulator(method='statevector')

                if (extra == 0):
                    special_had_test(gate_set[i], [1, 2, 3], 0, parameters,
                if (extra == 1):
                    special_had_test(gate_set[j], [1, 2, 3], 0, parameters,

                circ.save_statevector()
                t_circ = transpile(circ, backend)
                job = backend.run(t_circ, shots=10000)

                result = job.result()
                outputstate = np.real(result.get_statevector(circ, decimals=
                o = outputstate

                m_sum = 0
                for l in range (0, len(o)):
                    if (l%2 == 1):
                        n = o[l]**2
                        m_sum+=n
                mult = mult*(1-(2*m_sum))

            overall_sum_2+=multiply*mult

    #print(1-float(overall_sum_2/overall_sum_1))

    return 1-float(overall_sum_2/overall_sum_1)
```

Equation

$$A = 0.45Z_3 + 0.55I$$

In [52]:
```
coefficient_set = [0.55, 0.45]
gate_set = [[0, 0, 0], [0, 0, 1]]
```

```
out = minimize(calculate_cost_function, x0=[float(random.randint(0,3000))/10
```

```
Optimization terminated successfully    (Exit mode 0)
            Current function value: 1.9279130702987146e-07
            Iterations: 23
            Function evaluations: 244
            Gradient evaluations: 23
```

In [53]:
```
print(out)
out_f = [out['x'][0:3], out['x'][3:6], out['x'][6:9]]
print(out_f)
```

```
 message: Optimization terminated successfully
 success: True
  status: 0
     fun: 1.9279130702987146e-07
       x: [ 1.119e-01 -5.238e-04  4.518e+00 -7.057e-02  4.264e-04
            2.635e+00  1.713e+00  1.571e+00  2.068e+00]
     nit: 23
     jac: [ 6.203e-04 -1.267e-03  3.008e-04  7.435e-04 -1.118e-03
            2.009e-04 -1.007e-04 -9.976e-05  3.556e-04]
    nfev: 244
    njev: 23
[array([ 1.11923173e-01, -5.23763159e-04,  4.51828720e+00]), array([-7.05723
026e-02,  4.26407977e-04,  2.63495533e+00]), array([1.71328961, 1.57100673,
2.0683348 ])]
```

In [54]:
```
circ = QuantumCircuit(3, 3)
apply_fixed_ansatz([0, 1, 2], out_f)
circ.save_statevector()

backend = AerSimulator(method='statevector')
t_circ = transpile(circ, backend)
job = backend.run(t_circ, shots=10000)

result = job.result()
o = result.get_statevector(circ, decimals=10)

a1 = coefficient_set[1]*np.array([[1,0,0,0,0,0,0,0], [0,1,0,0,0,0,0,0], [0,0
a2 = coefficient_set[0]*np.array([[1,0,0,0,0,0,0,0], [0,1,0,0,0,0,0,0], [0,0
a3 = np.add(a1, a2)

b = np.array([float(1/np.sqrt(8)),float(1/np.sqrt(8)),float(1/np.sqrt(8)),fl
```

In [55]:
```
print("Solver Accuracy")
print((b.dot(a3.dot(o)/(np.linalg.norm(a3.dot(o)))))**2)
```

```
Solver Accuracy
(0.9999998072085439-0j)
```

In [56]:
```
print("a1")
print(a1)
print("a2")
print(a2)
print("a3")
print(a3)
```

```
print("b")
print(b)
```

```
a1
[[ 0.45  0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.45  0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.45  0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.45  0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.   -0.45  0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.   -0.45  0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.   -0.45  0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.   -0.45]]
a2
[[0.55 0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.55 0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.55 0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.55 0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.55 0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.55 0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.55 0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.55]]
a3
[[1.  0.  0.  0.  0.  0.  0.  0. ]
 [0.  1.  0.  0.  0.  0.  0.  0. ]
 [0.  0.  1.  0.  0.  0.  0.  0. ]
 [0.  0.  0.  1.  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.1 0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.1 0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.1 0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.1]]
b
[0.35355339 0.35355339 0.35355339 0.35355339 0.35355339 0.35355339
 0.35355339 0.35355339]
```

Equation

$$A = 0.55I + 0.225Z_2 + 0.225Z_3$$

In [47]:
```
coefficient_set = [0.55, 0.225, 0.225]
gate_set = [[0, 0, 0], [0, 1, 0], [0, 0, 1]]

out = minimize(calculate_cost_function, x0=[float(random.randint(0,3000))/10
```

```
Optimization terminated successfully    (Exit mode 0)
            Current function value: 6.209547809277183e-08
            Iterations: 20
            Function evaluations: 206
            Gradient evaluations: 20
```

In [48]:
```
print(out)
out_f = [out['x'][0:3], out['x'][3:6], out['x'][6:9]]
print(out_f)
```

```
 message: Optimization terminated successfully
 success: True
  status: 0
     fun: 6.209547809277183e-08
       x: [ 2.737e+00 -5.125e-04  1.453e+00  1.852e+00  3.164e-01
            1.351e+00  2.929e+00  2.465e+00  3.150e+00]
     nit: 20
     jac: [ 7.413e-06 -1.630e-04  5.960e-04 -1.128e-04 -9.649e-05
           -5.947e-04  1.276e-04  9.032e-05 -2.771e-04]
    nfev: 206
    njev: 20
[array([ 2.73724692e+00, -5.12474150e-04,  1.45252565e+00]), array([1.852174
61, 0.31644919, 1.3506257 ]), array([2.9294482 , 2.46545151, 3.15028422])]
```

In [49]:
```
circ = QuantumCircuit(3, 3)
apply_fixed_ansatz([0, 1, 2], out_f)
circ.save_statevector()

backend = AerSimulator(method='statevector')

t_circ = transpile(circ, backend)
job = backend.run(t_circ, shots=10000)

result = job.result()
o = result.get_statevector(circ, decimals=10)

a1 = coefficient_set[2]*np.array([[1,0,0,0,0,0,0,0], [0,1,0,0,0,0,0,0], [0,0
a0 = coefficient_set[1]*np.array([[1,0,0,0,0,0,0,0], [0,1,0,0,0,0,0,0], [0,0
a2 = coefficient_set[0]*np.array([[1,0,0,0,0,0,0,0], [0,1,0,0,0,0,0,0], [0,0

a3 = np.add(np.add(a2, a0), a1)

b = np.array([float(1/np.sqrt(8)),float(1/np.sqrt(8)),float(1/np.sqrt(8)),fl
```

In [50]:
```
print("Solver Accuracy")
print((b.dot(a3.dot(o)/(np.linalg.norm(a3.dot(o)))))**2)
```

```
Solver Accuracy
(0.999999937904522-0j)
```

In [51]:
```
print("a0")
print(a0)
print("a1")
print(a1)
print("a2")
print(a2)
print("a3")
print(a3)
print("b")
print(b)
```

```
a0
[[ 0.225  0.     0.     0.     0.     0.     0.     0.   ]
 [ 0.     0.225  0.     0.     0.     0.     0.     0.   ]
 [ 0.     0.    -0.225  0.     0.     0.     0.     0.   ]
 [ 0.     0.     0.    -0.225  0.     0.     0.     0.   ]
 [ 0.     0.     0.     0.     0.225  0.     0.     0.   ]
 [ 0.     0.     0.     0.     0.     0.225  0.     0.   ]
 [ 0.     0.     0.     0.     0.     0.    -0.225  0.   ]
 [ 0.     0.     0.     0.     0.     0.     0.    -0.225]]
a1
[[ 0.225  0.     0.     0.     0.     0.     0.     0.   ]
 [ 0.     0.225  0.     0.     0.     0.     0.     0.   ]
 [ 0.     0.     0.225  0.     0.     0.     0.     0.   ]
 [ 0.     0.     0.     0.225  0.     0.     0.     0.   ]
 [ 0.     0.     0.     0.    -0.225  0.     0.     0.   ]
 [ 0.     0.     0.     0.     0.    -0.225  0.     0.   ]
 [ 0.     0.     0.     0.     0.     0.    -0.225  0.   ]
 [ 0.     0.     0.     0.     0.     0.     0.    -0.225]]
a2
[[0.55 0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.55 0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.55 0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.55 0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.55 0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.55 0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.55 0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.55]]
a3
[[1.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   1.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.55 0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.55 0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.55 0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.55 0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.1  0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.1 ]]
b
[0.35355339 0.35355339 0.35355339 0.35355339 0.35355339 0.35355339
 0.35355339 0.35355339]
```

In [ ]: