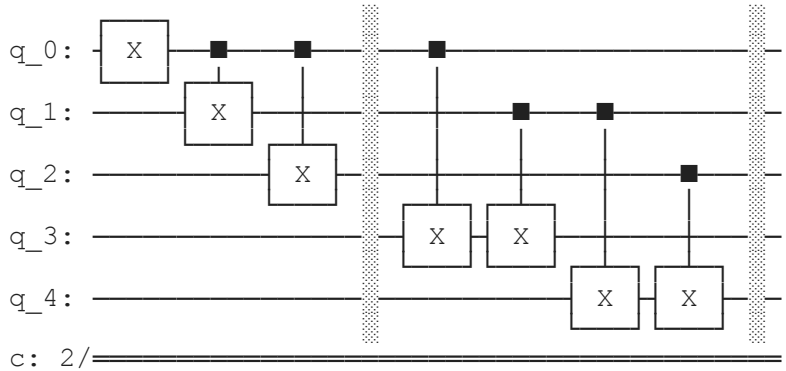
```
!pip install qiskit
!pip install git+https://github.com/qiskit-community/qiskit-textbook.git#subdirectory
!pip install numexpr
!pip install pylatexenc

# Do the necessary imports
import numpy as np
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit import IBMQ, Aer, transpile, assemble, execute
from qiskit.visualization import plot_histogram, plot_bloch_multivector
from qiskit.extensions import Initialize
from qiskit_textbook.tools import random_state, array_to_latex
from math import sqrt, pi
from qiskit.quantum_info import *
from qiskit.visualization import *
from qiskit.result import *
```

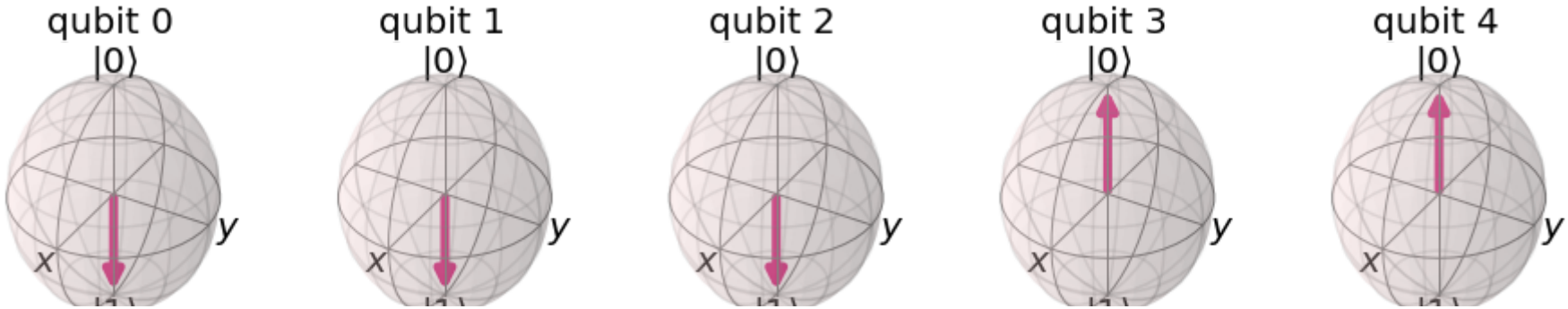
1.- CREATING THE 3 BIT FLIP-CODE CIRCUIT

```
def encoding(qc, d0, d1, d2):

    #### your code goes here
    qc.x(d0)
    qc.cnot(d0,d1)
    qc.cnot(d0,d2)
    qc.barrier()


def detection(qc, d0, d1, d2, a0, a1):

     #### your code goes here

    qc.cnot(d0,a0)
```

```
    qc.cnot(d1,a0)
    qc.cnot(d1,a1)
    qc.cnot(d2,a1)

    qc.barrier()

    #measurement ancilla qubits
    #### your code goes here

## SETUP
bitflip_circuit = QuantumCircuit(5,2)

## STEP 1: encoding
encoding(bitflip_circuit, 0,1,2)


## STEP 2:detection
detection(bitflip_circuit,0,1,2,3,4)

bitflip_circuit.draw()
```




```
sv_sim = Aer.get_backend('statevector_simulator')
qobj = assemble(bitflip_circuit)
out_vector = sv_sim.run(qobj).result().get_statevector()
plot_bloch_multivector(out_vector)
```



CHECKING THE RESULTS:

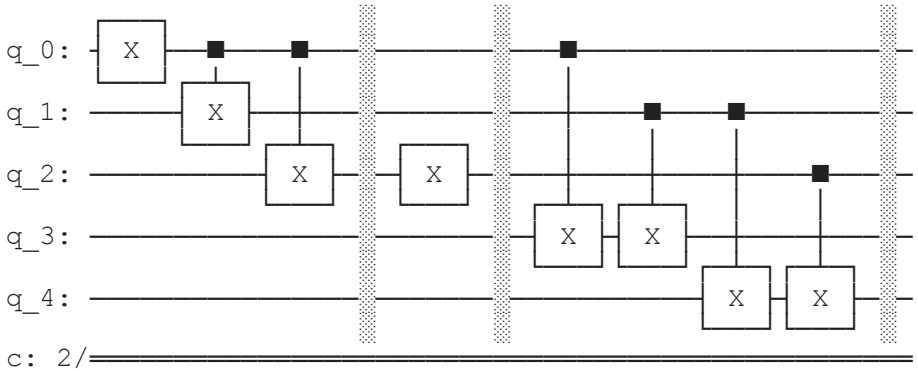
a) What are the states of q3 and q4 (ancilla qubits) and why?

b) What are the states of $|q_0\rangle$, $|q_1\rangle$ and $|q_2\rangle$? Or in other words, what is the state of the logical qubit $|\psi_L\rangle = |q_2 q_1 q_0\rangle$?

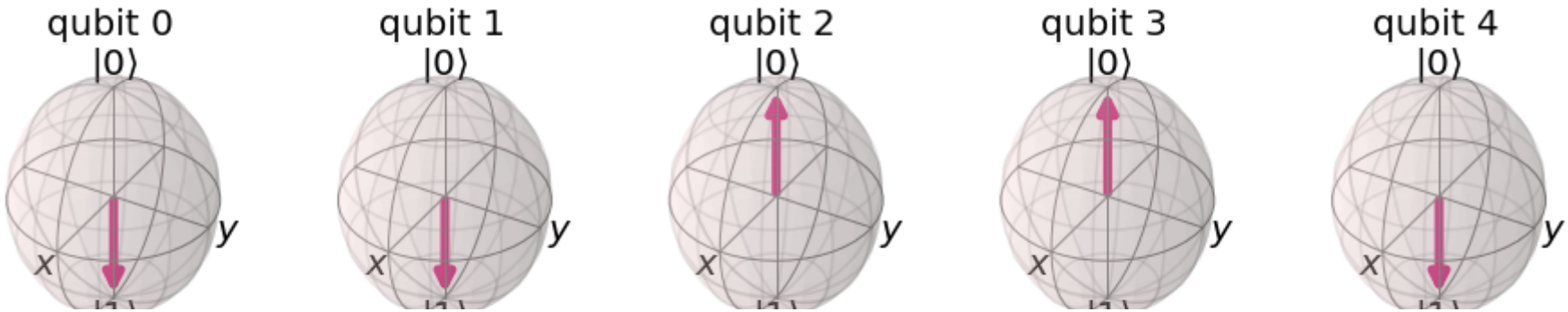
2.- INJECTING BIT-FLIP ERRORS AND DECODING THEM

```
import random

def error_injection(qc, d0, d1, d2):
    #function for injecting a bit-flip on q0, or on q1 or on q2
    rand = random.randint(1,3)
    #### your code goes here
    if rand == 1: qc.x(d0)
    if rand == 2: qc.x(d1)
    if rand == 3: qc.x(d2)
    qc.barrier()


## SETUP

bitflip_circuit = QuantumCircuit(5,2)
```

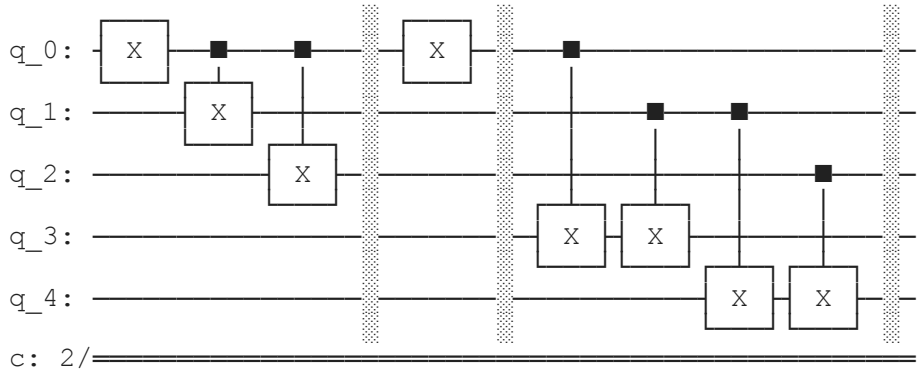
```
## STEP 1: encoding
encoding(bitflip_circuit, 0,1,2)

## STEP 2: inserting errors
error_injection(bitflip_circuit, 0,1,2)

## STEP 3:detection
detection(bitflip_circuit,0,1,2,3,4)

bitflip_circuit.draw()
```




```
sv_sim = Aer.get_backend('statevector_simulator')
qobj = assemble(bitflip_circuit)
out_vector = sv_sim.run(qobj).result().get_statevector()
plot_bloch_multivector(out_vector)
```

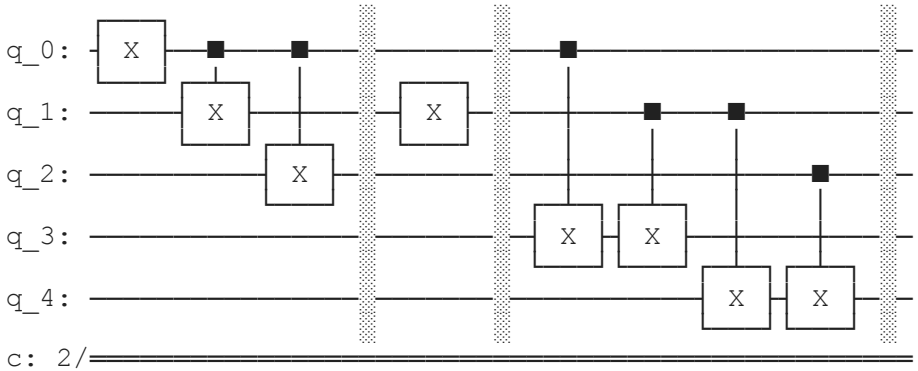


```
## SETUP

bitflip_circuit = QuantumCircuit(5,2)

## STEP 1: encoding
encoding(bitflip_circuit, 0,1,2)

## STEP 2: inserting errors
error_injection(bitflip_circuit, 0,1,2)

## STEP 3:detection
detection(bitflip_circuit,0,1,2,3,4)

bitflip_circuit.draw()
```

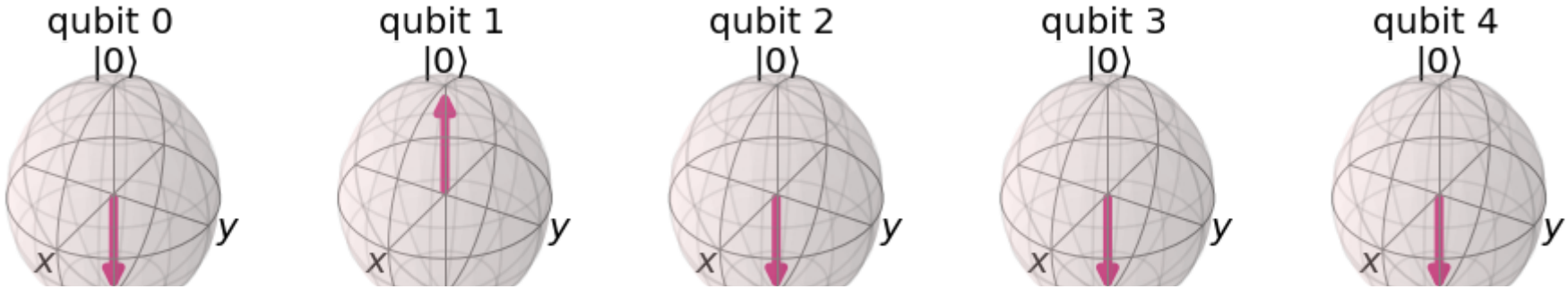



```
sv_sim = Aer.get_backend('statevector_simulator')
qobj = assemble(bitflip_circuit)
out_vector = sv_sim.run(qobj).result().get_statevector()
plot_bloch_multivector(out_vector)
```


```
## SETUP

bitflip_circuit = QuantumCircuit(5,2)

## STEP 1: encoding
encoding(bitflip_circuit, 0,1,2)

## STEP 2: inserting errors
error_injection(bitflip_circuit, 0,1,2)

## STEP 3:detection
detection(bitflip_circuit,0,1,2,3,4)

bitflip_circuit.draw()
```




```
sv_sim = Aer.get_backend('statevector_simulator')
qobj = assemble(bitflip_circuit)
out_vector = sv_sim.run(qobj).result().get_statevector()
plot_bloch_multivector(out_vector)
```

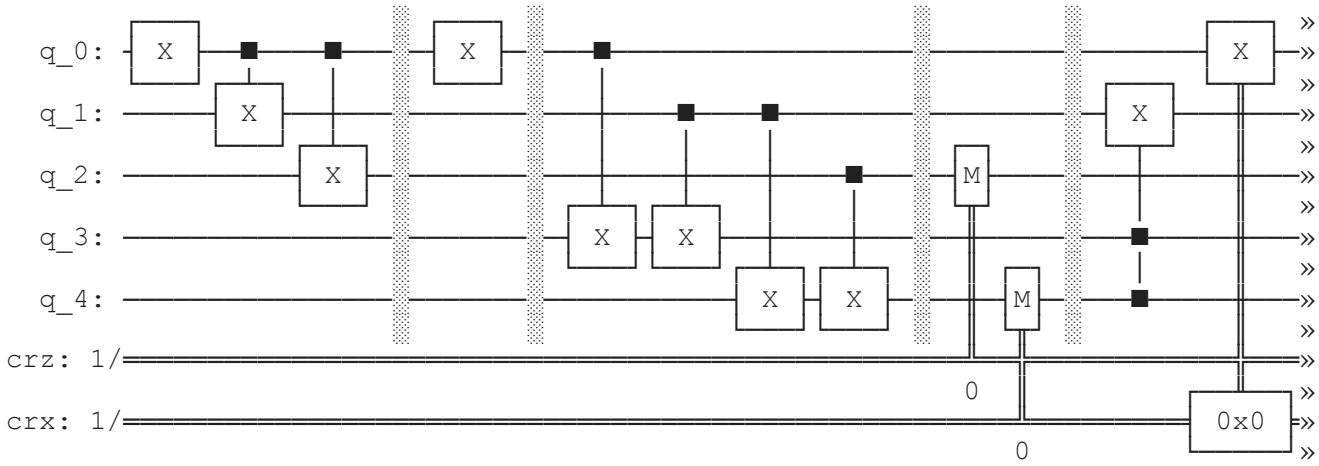


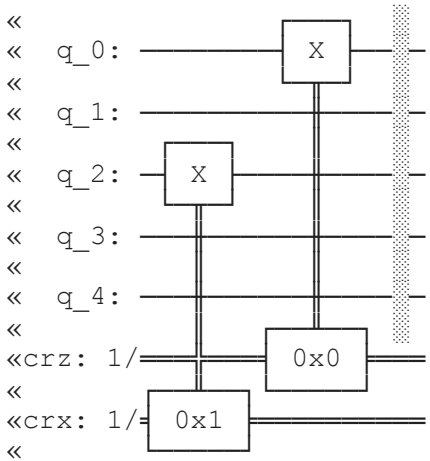
Error (Bit-flip)	Logical state $\|\psi_L\rangle = \|q_2 q_1 q_0\rangle$	Syndrome $\|q_4 q_3\rangle$
q0	|110⟩	|01⟩
q1	|001⟩	|11⟩
q2	|011⟩	|10⟩

3.- CORRECTING BIT-FLIP ERRORS

```
import random

def error_injection(qc, d0, d1, d2):
    #function for injecting a bit-flip on q0, or on q1 or on q2
    rand = random.randint(1,3)
    #### your code goes here
    if rand == 1: qc.x(d0)
    if rand == 2: qc.x(d1)
    if rand == 3: qc.x(d2)
    qc.barrier()
```

```
def correction(qc, d0, d1, d2, d3, d4, a0, a1):
     #### your code goes here

    qc.measure(d3,a0)
    qc.measure(d4,a1)

    qc.barrier()

    #correction d1
    qc.ccx(d3,d4,d1)

    #correction do y d2
    qc.x(d0).c_if(a1,0)
    qc.x(d2).c_if(a1,1)
    qc.x(d0).c_if(a0,0)

    qc.barrier()

## SETUP
qr = QuantumRegister(5, name="q")
crz, crx = ClassicalRegister(1, name="crz"), ClassicalRegister(1, name="crx")
bitflip_circuit = QuantumCircuit(qr,crz,crx)

## STEP 1: encoding
encoding(bitflip_circuit, 0,1,2)

## STEP 2: inserting errors
error_injection(bitflip_circuit, 0,1,2)

## STEP 3:detection
detection(bitflip_circuit,0,1,2,3,4)

## STEP 4:correction
correction(bitflip_circuit,0,1,2,3,4,crz,crx)


bitflip_circuit.draw()
```



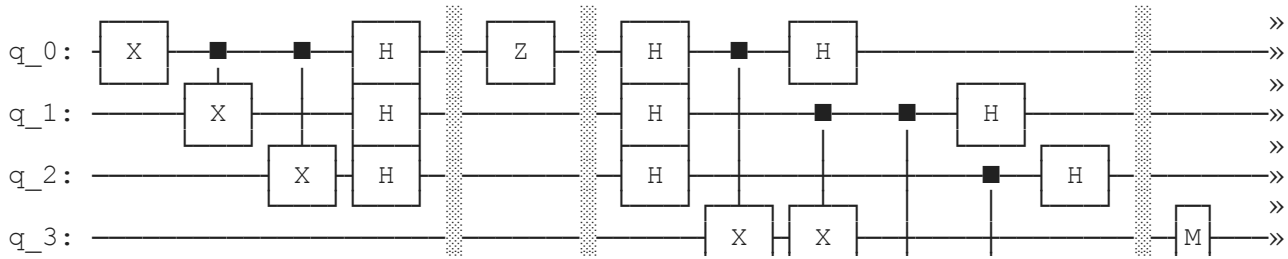


THE 3 QUBIT PHASE-FLIP CODE

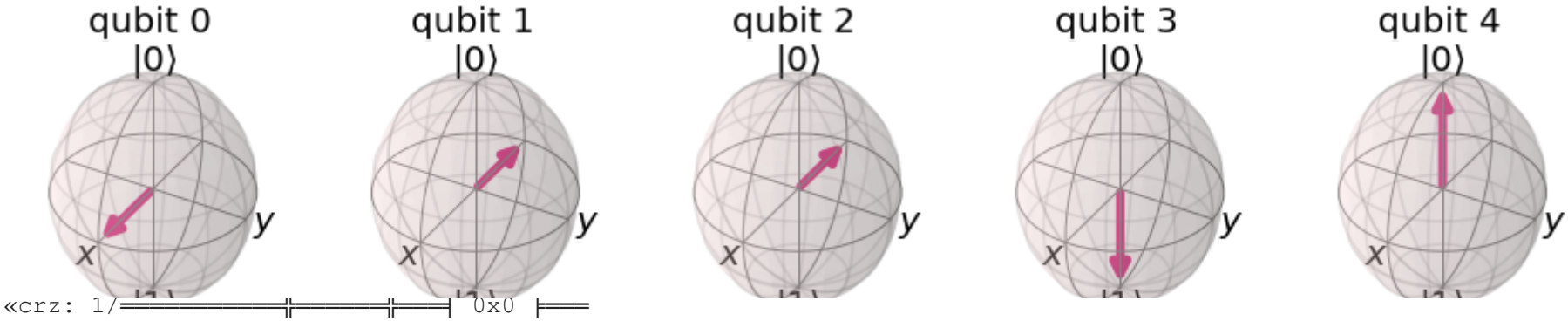
```
def encoding(qc, d0, d1, d2):

    #### your code goes here
    qc.x(d0)
    qc.cnot(d0,d1)
    qc.cnot(d0,d2)
    qc.h(d0)
```

```
    qc.h(d1)
    qc.h(d2)
    qc.barrier()


import random

def error_injection(qc, d0, d1, d2):
    #function for injecting a bit-flip on q0, or on q1 or on q2
    rand = random.randint(1,3)
    #### your code goes here
    if rand == 1: qc.z(d0)
    if rand == 2: qc.z(d1)
    if rand == 3: qc.z(d2)
    qc.barrier()


def detection(qc, d0, d1, d2,a0,a1):

     #### your code goes here
    qc.h(d0)
    qc.h(d1)
    qc.h(d2)
    qc.cnot(d0,a0)
    qc.cnot(d1,a0)
    qc.cnot(d1,a1)
    qc.cnot(d2,a1)
    qc.h(d0)
    qc.h(d1)
    qc.h(d2)

    qc.barrier()

    #measurement ancilla qubit



## SETUP
```

```
qr = QuantumRegister(5, name="q")
crz, crx = ClassicalRegister(1, name="crz"), ClassicalRegister(1, name="crx")
phaseflip_circuit = QuantumCircuit(qr,crz,crx)

## STEP 1: encoding
encoding(phaseflip_circuit, 0,1,2)

## STEP 2: inserting errors
error_injection(phaseflip_circuit, 0,1,2)

## STEP 3:detection
detection(phaseflip_circuit,0,1,2,3,4)

## STEP 4:correction
correction(phaseflip_circuit,0,1,2,3,4,crz,crx)


phaseflip_circuit.draw()
```




```
sv_sim = Aer.get_backend('statevector_simulator')
qobj = assemble(phaseflip_circuit)
out_vector = sv_sim.run(qobj).result().get_statevector()
plot_bloch_multivector(out_vector)
```



THE 7 QUBIT STEANE CODE

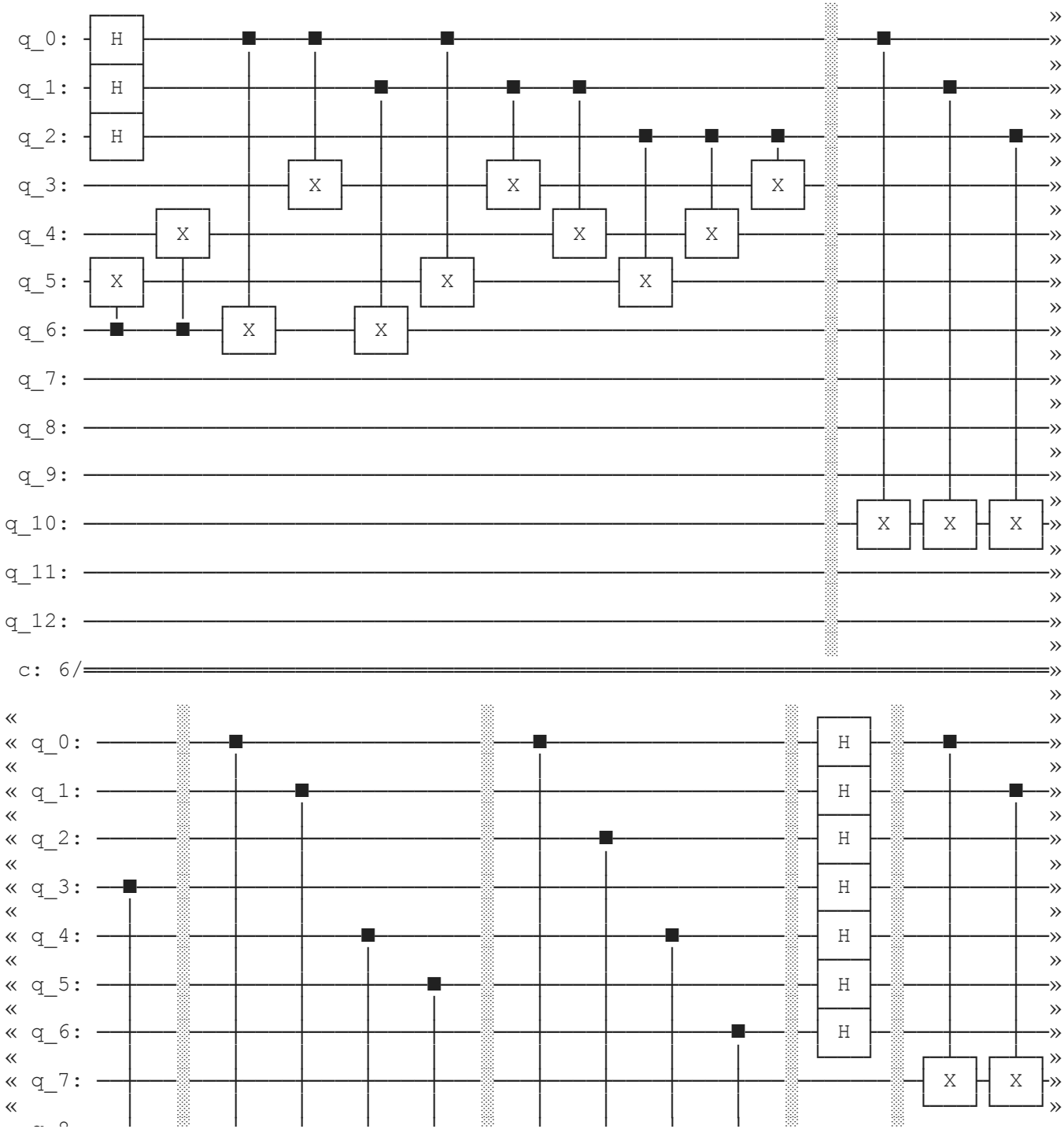
```
def encoding(qc, d0, d1, d2, d3, d4, d5, d6):
  qc.h(d0)
  qc.h(d1)
  qc.h(d2)
  qc.cnot(d6,d5)
  qc.cnot(d6,d4)
  qc.cnot(d0,d6)
  qc.cnot(d0,d3)
  qc.cnot(d0,d5)
  qc.cnot(d1,d6)
```

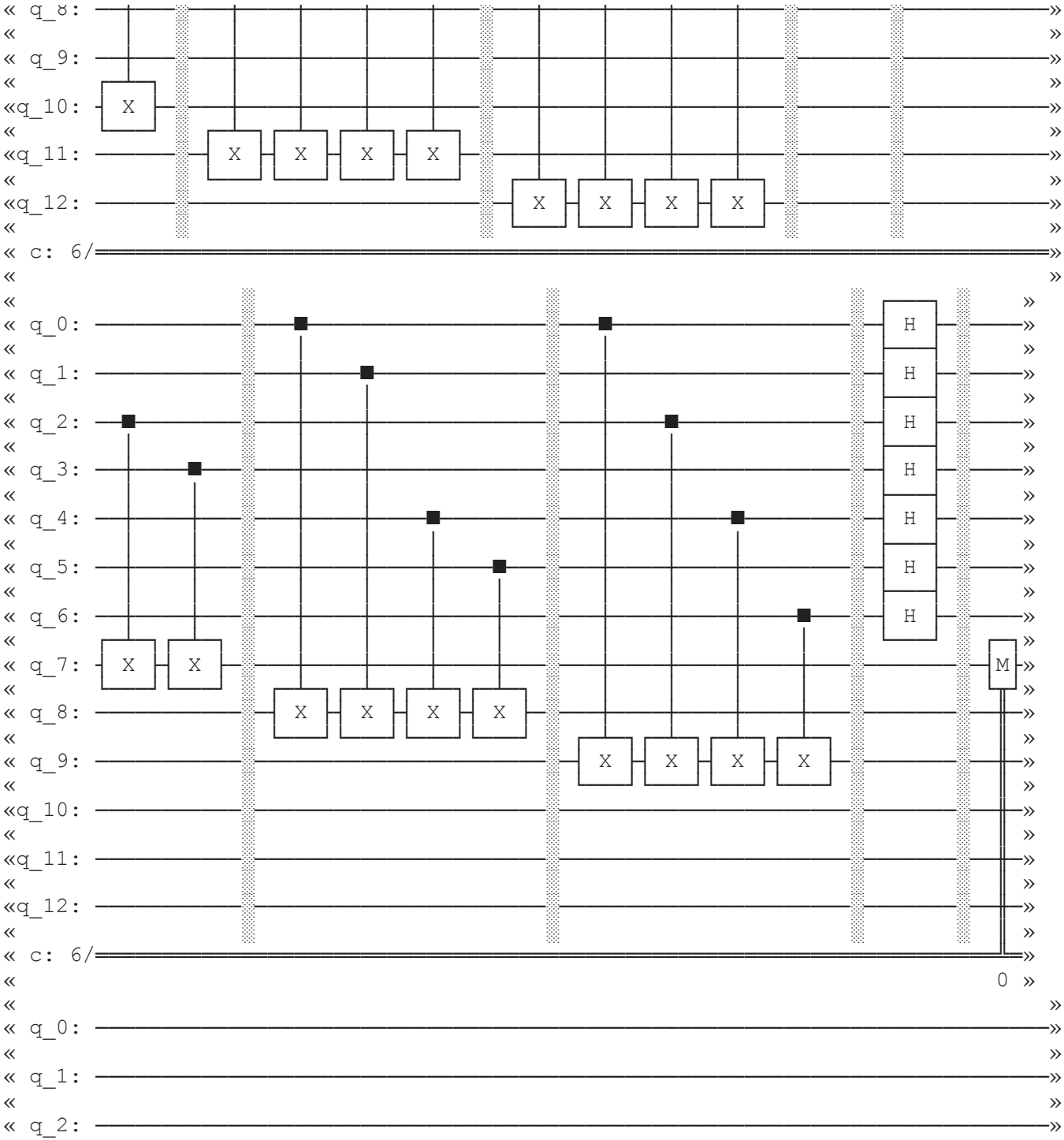
```
    qc.cnot(d1,d3)
    qc.cnot(d1,d4)
    qc.cnot(d2,d5)
    qc.cnot(d2,d4)
    qc.cnot(d2,d3)
    qc.barrier()

import random

def error_injection(qc, d0, d1, d2, d3, d4, d5, d6):
#function for injecting  phase-flip  and bit-flip errors
    rand = random.randint(1,7)
    rand2 = random.randint(1,2)
    #### your code goes here
    if rand == 1 and rand2 == 1: qc.x(d0)
    if rand == 1 and rand2 == 2: qc.z(d0)

    if rand == 2 and rand2 == 1: qc.x(d1)
    if rand == 2 and rand2 == 2: qc.z(d1)

    if rand == 3 and rand2 == 1: qc.x(d2)
    if rand == 3 and rand2 == 2: qc.z(d2)

    if rand == 4 and rand2 == 1: qc.x(d3)
    if rand == 4 and rand2 == 2: qc.z(d3)

    if rand == 5 and rand2 == 1: qc.x(d4)
    if rand == 5 and rand2 == 2: qc.z(d4)

    if rand == 6 and rand2 == 1: qc.x(d5)
    if rand == 6 and rand2 == 2: qc.z(d5)

    if rand == 7 and rand2 == 1: qc.x(d6)
    if rand == 7 and rand2 == 2: qc.z(d6)


    qc.barrier()
```

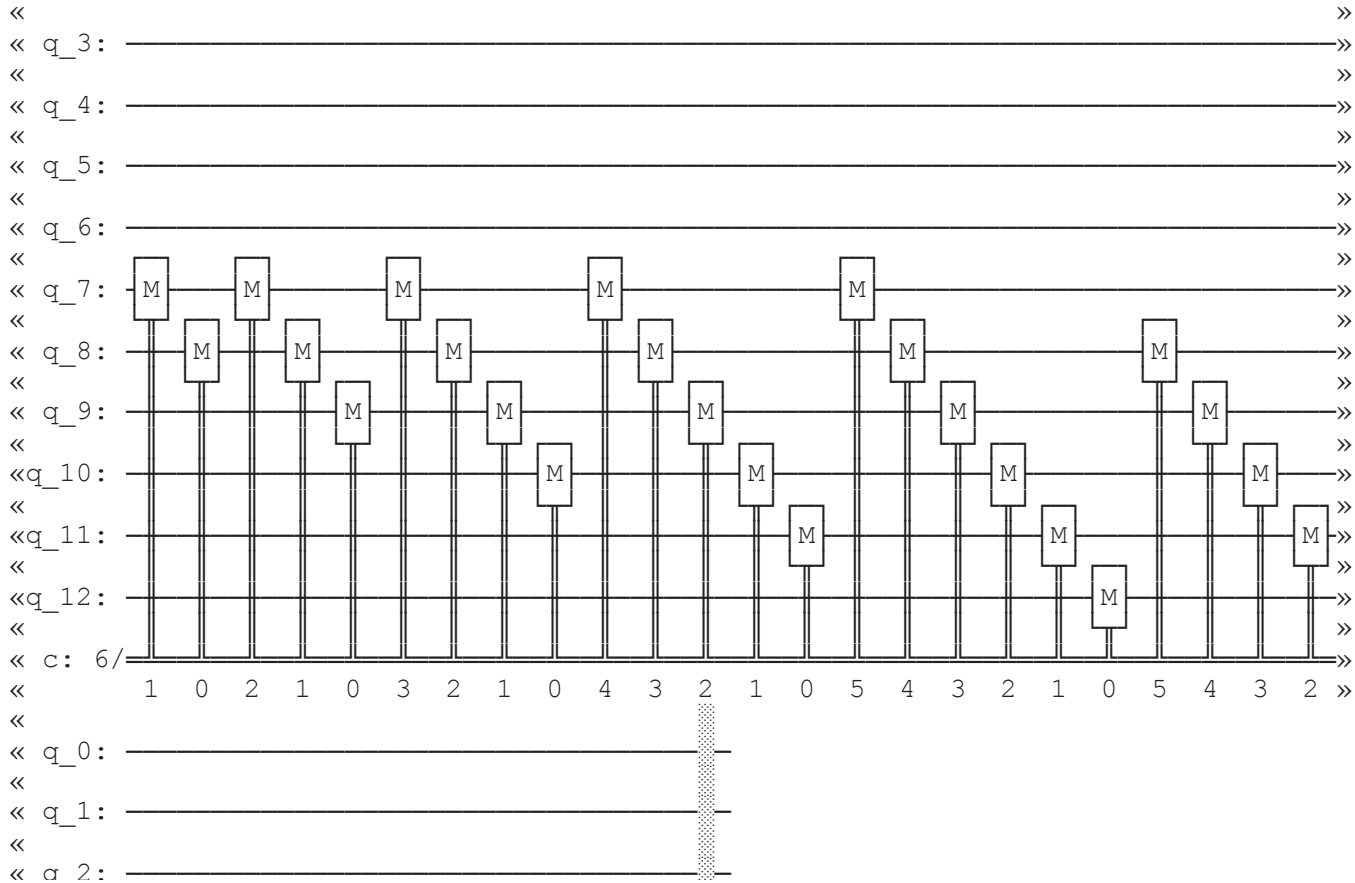
```
def detection(qc,d0,d1,d2,d3,d4,d5,d6,a1,a2,a3,a4,a5,a6,c):
  #ancilla 1
  qc.cnot(d0,a4)
  qc.cnot(d1,a4)
  qc.cnot(d2,a4)
  qc.cnot(d3,a4)
  qc.barrier()

  #ancilla 2
  qc.cnot(d0,a5)
  qc.cnot(d1,a5)
  qc.cnot(d4,a5)
  qc.cnot(d5,a5)
  qc.barrier()

  #ancilla 3
  qc.cnot(d0,a6)
  qc.cnot(d2,a6)
  qc.cnot(d4,a6)
  qc.cnot(d6,a6)
  qc.barrier()

  #hadamard
  qc.h(d0)
  qc.h(d1)
  qc.h(d2)
  qc.h(d3)
  qc.h(d4)
  qc.h(d5)
  qc.h(d6)
  qc.barrier()

  #ancilla 4
  qc.cnot(d0,a1)
  qc.cnot(d1,a1)
  qc.cnot(d2,a1)
  qc.cnot(d3,a1)
```

```
qc.barrier()

#ancilla 5
qc.cnot(d0,a2)
qc.cnot(d1,a2)
qc.cnot(d4,a2)
qc.cnot(d5,a2)
qc.barrier()

#ancilla 6
qc.cnot(d0,a3)
qc.cnot(d2,a3)
qc.cnot(d4,a3)
qc.cnot(d6,a3)
qc.barrier()

#hadamard
qc.h(d0)
qc.h(d1)
qc.h(d2)
qc.h(d3)
qc.h(d4)
qc.h(d5)
qc.h(d6)
qc.barrier()


#MEASUREMENT

qc.measure(a1,c)
qc.measure(a2,c)
qc.measure(a3,c)
qc.measure(a4,c)
qc.measure(a5,c)
qc.measure(a6,c)
qc.barrier()
```

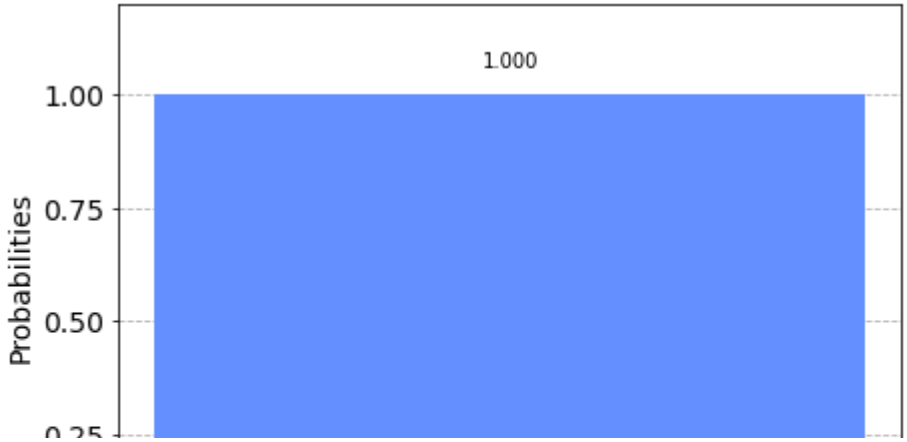
```
qr = QuantumRegister(13, name="q")
cr = ClassicalRegister(6, name="c")
## SETUP
steane_circuit = QuantumCircuit(qr,cr)

## STEP 1: encoding
encoding(steane_circuit,0,1,2,3,4,5,6)

## STEP 2: inserting errors
#error_injection(steane_circuit,0,1,2,3,4,5,6)

## STEP 3:detection
detection(steane_circuit,0,1,2,3,4,5,6,7,8,9,10,11,12,cr)

steane_circuit.draw()
```


q_0:
q_1:
q_2:
q_3:
q_4:
q_5:
q_6:
q_7:
q_8:
q_9:
q_10:
q_11:
q_12:
c: 6/
H
X


« q_8:
« q_9:
«q_10:
«q_11:
«q_12:
« c: 6/
« q_0:
« q_1:
« q_2:
« q_3:
« q_4:
« q_5:
« q_6:
« q_7:
« q_8:
« q_9:
«q_10:
«q_11:
«q_12:
« c: 6/
0
« q_0:
« q_1:
« q_2:
X
H
M

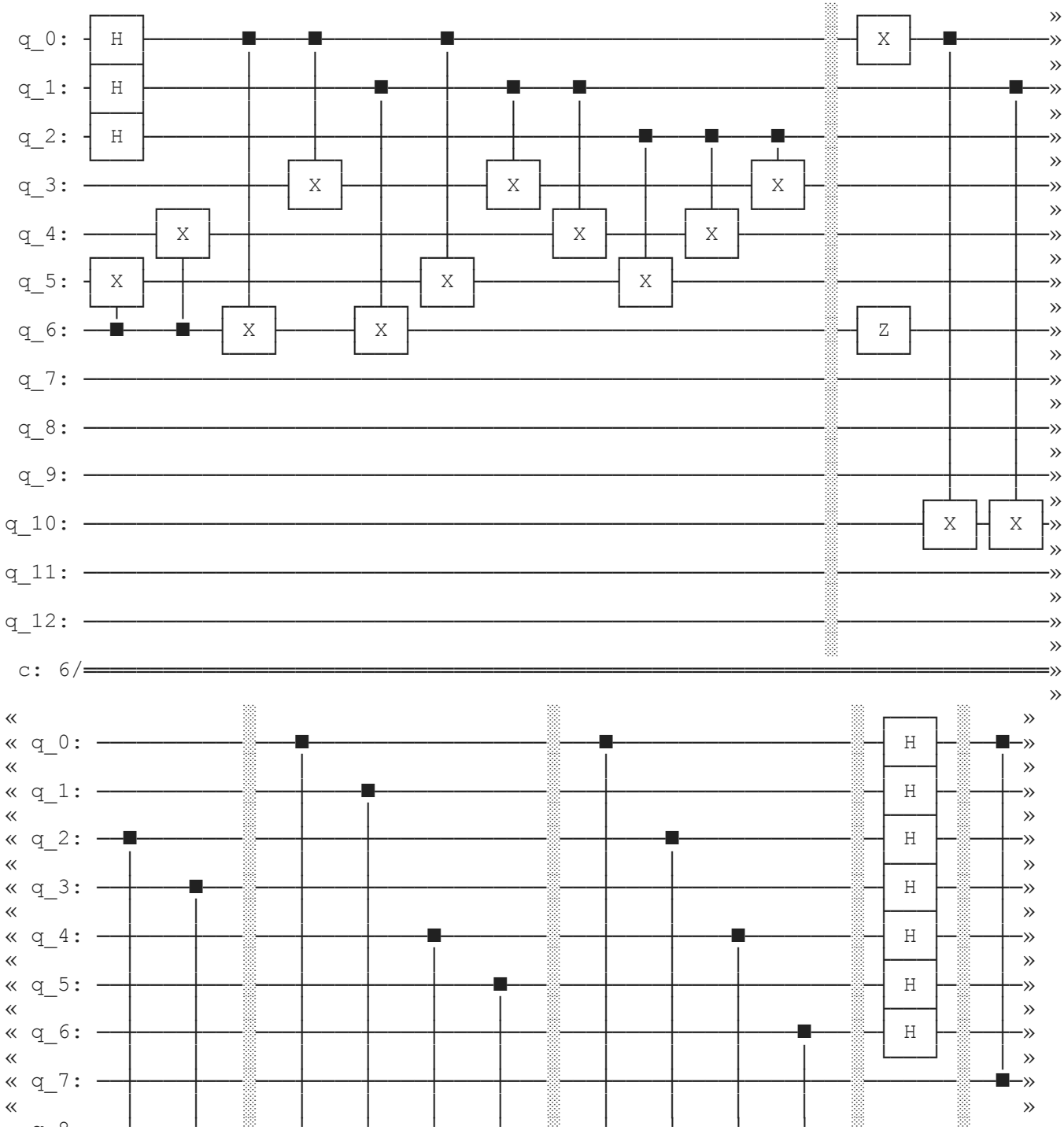



```
qasm_sim = Aer.get_backend('qasm_simulator')
t_qc = transpile(steane_circuit, qasm_sim)
qobj = assemble(t_qc)
results = qasm_sim.run(qobj).result()
counts = results.get_counts()
plot_histogram(counts)
```

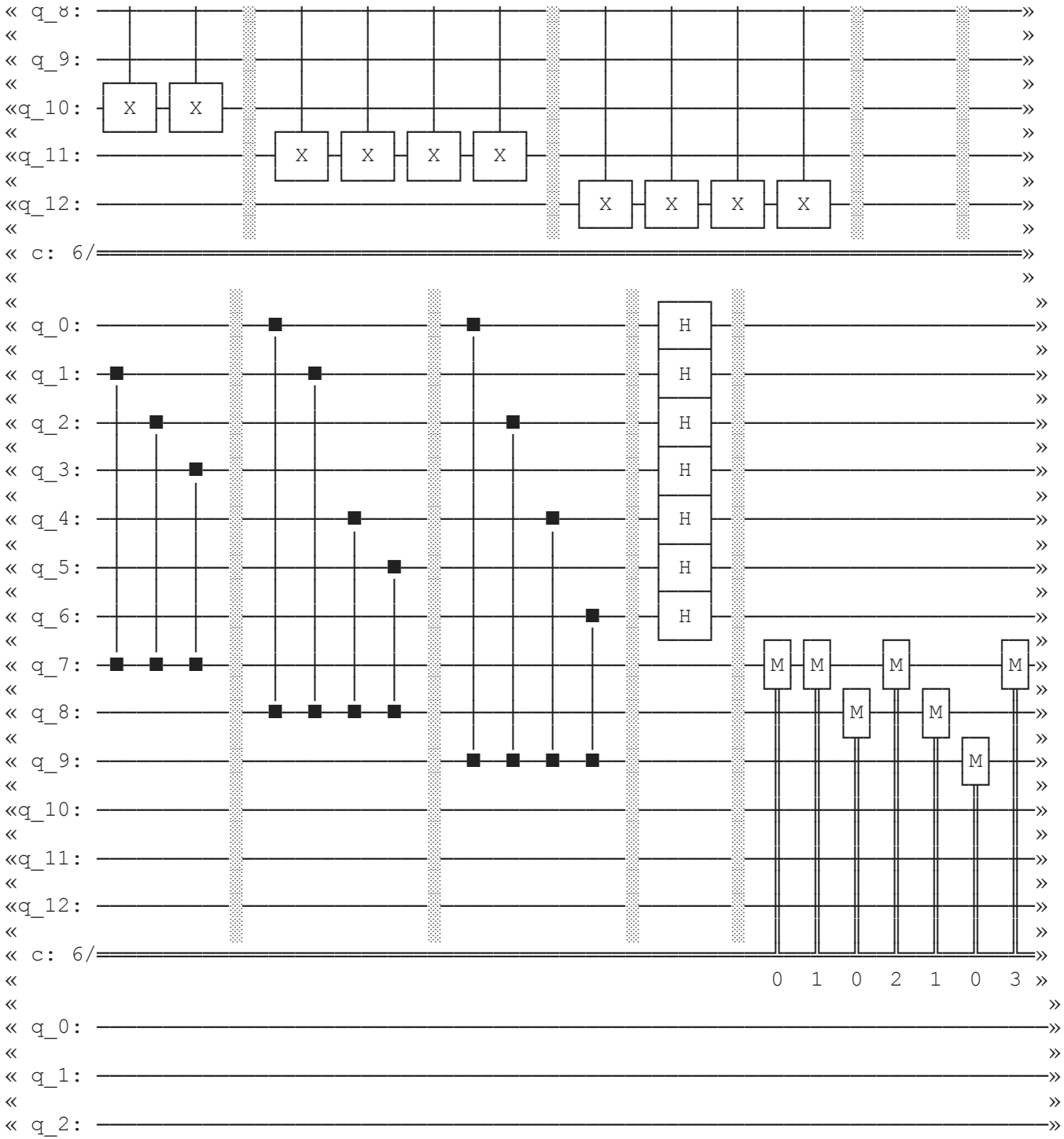


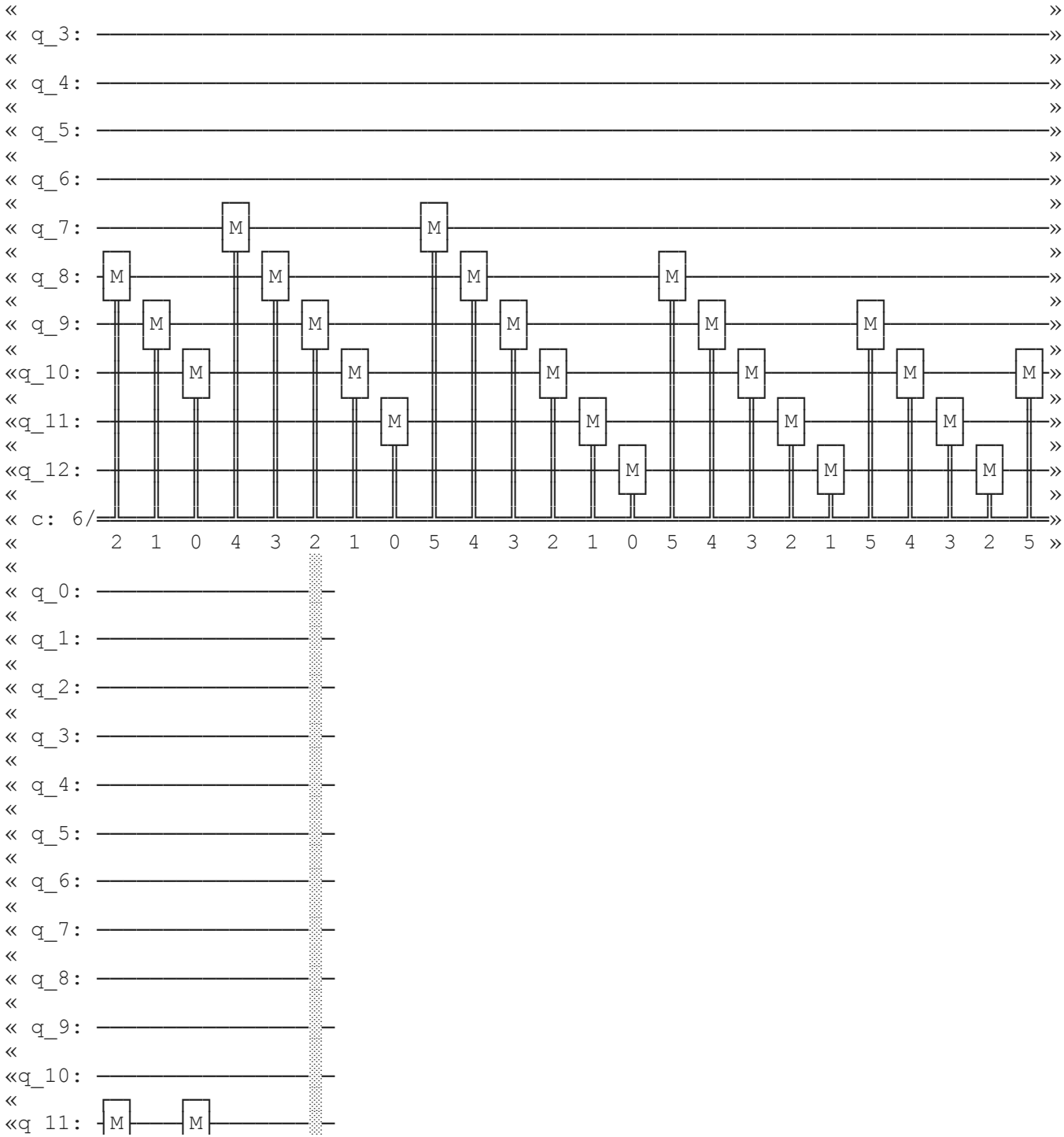
What is the error syndrome if we inject a bit-flip error on q0 and a phase flip error on q6? Are these errors detectable and correctable?

```
def error_injection(qc, d0, d1, d2, d3, d4, d5, d6):
  qc.x(d0)
  qc.z(d6)


qr = QuantumRegister(13, name="q")
cr = ClassicalRegister(6, name="c")
## SETUP
steane_circuit = QuantumCircuit(qr,cr)

## STEP 1: encoding
encoding(steane_circuit,0,1,2,3,4,5,6)

## STEP 2: inserting errors
error_injection(steane_circuit,0,1,2,3,4,5,6)

## STEP 3:detection
detection(steane_circuit,0,1,2,3,4,5,6,7,8,9,10,11,12,cr)

steane_circuit.draw()
```


q_0:
q_1:
q_2:
q_3:
q_4:
q_5:
q_6:
q_7:
q_8:
q_9:
q_10:
q_11:
q_12:
c: 6/
H
X
Z

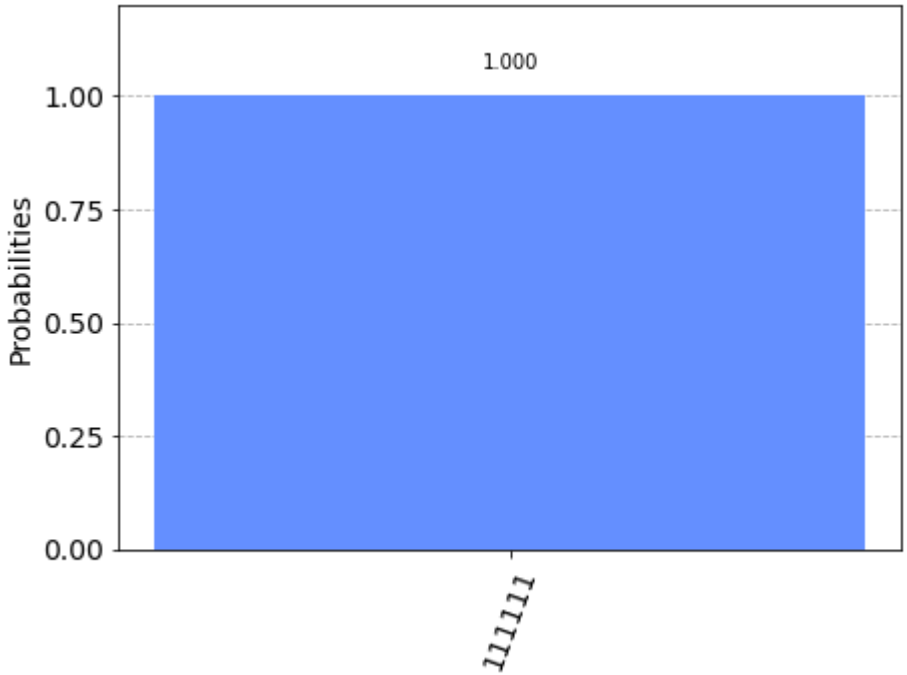
q_0:
q_1:
q_2:
q_3:
q_4:
q_5:
q_6:
q_7:
H


« q_8:
« q_9:
«q_10:
«q_11:
«q_12:
« c: 6/
X
« q_0:
« q_1:
« q_2:
« q_3:
« q_4:
« q_5:
« q_6:
« q_7:
« q_8:
« q_9:
«q_10:
«q_11:
«q_12:
« c: 6/
H
M
0 1 0 2 1 0 3
« q_0:
« q_1:
« q_2:


q_3:
q_4:
q_5:
q_6:
q_7:
q_8:
q_9:
q_10:
q_11:
q_12:
c: 6/
2 1 0 4 3 2 1 0 5 4 3 2 1 0 5 4 3 2 1 5 4 3 2 5
q_0:
q_1:
q_2:
q_3:
q_4:
q_5:
q_6:
q_7:
q_8:
q_9:
q_10:
q_11:
M

```
«                 
«q_12: ─┤M├─┤M├─┤M├─
«                 
« c: 6/═══════════════
```

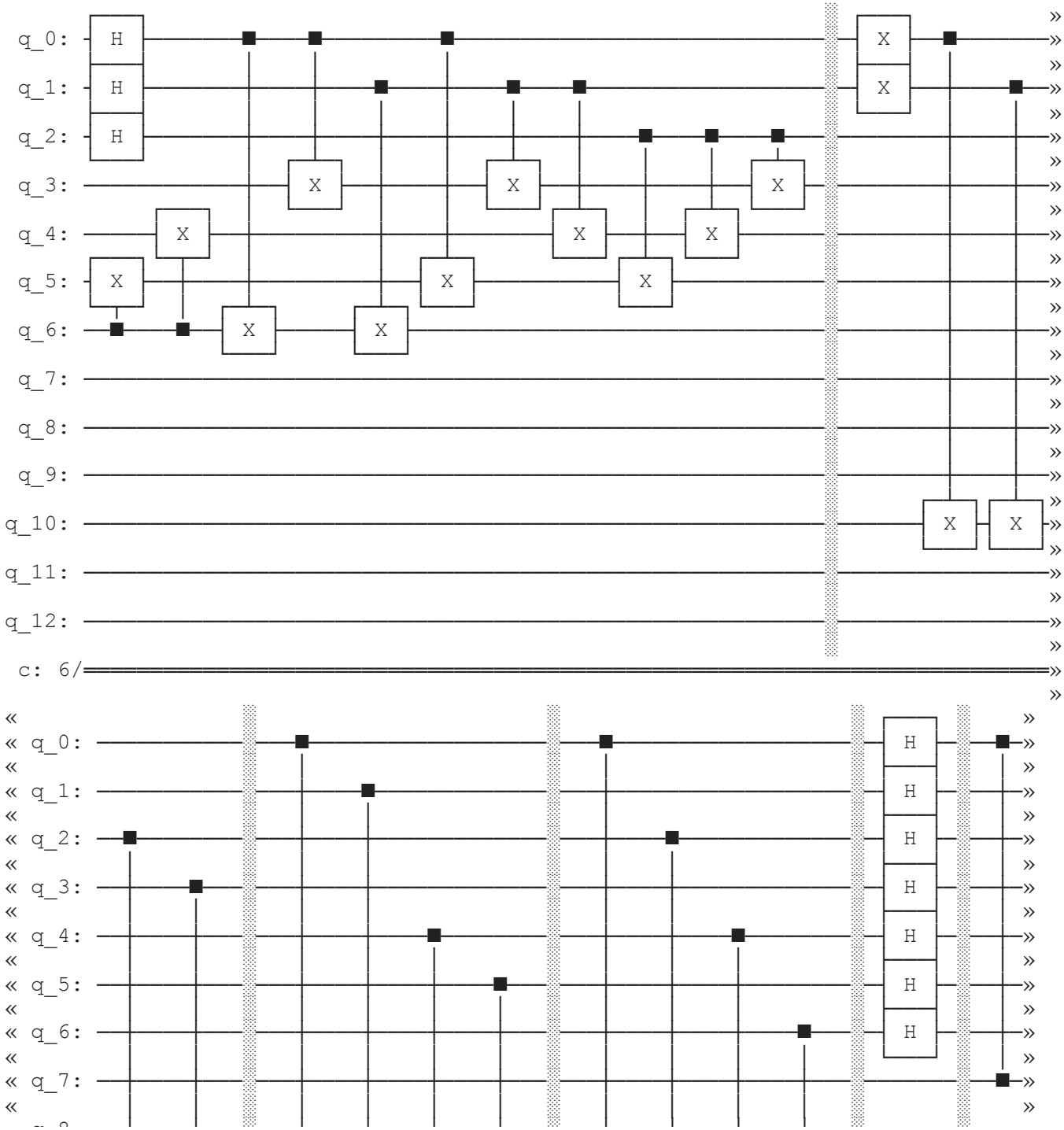
```
qasm_sim = Aer.get_backend('qasm_simulator')
t_qc = transpile(steane_circuit, qasm_sim)
qobj = assemble(t_qc)
results = qasm_sim.run(qobj).result()
counts = results.get_counts()
plot_histogram(counts)
```

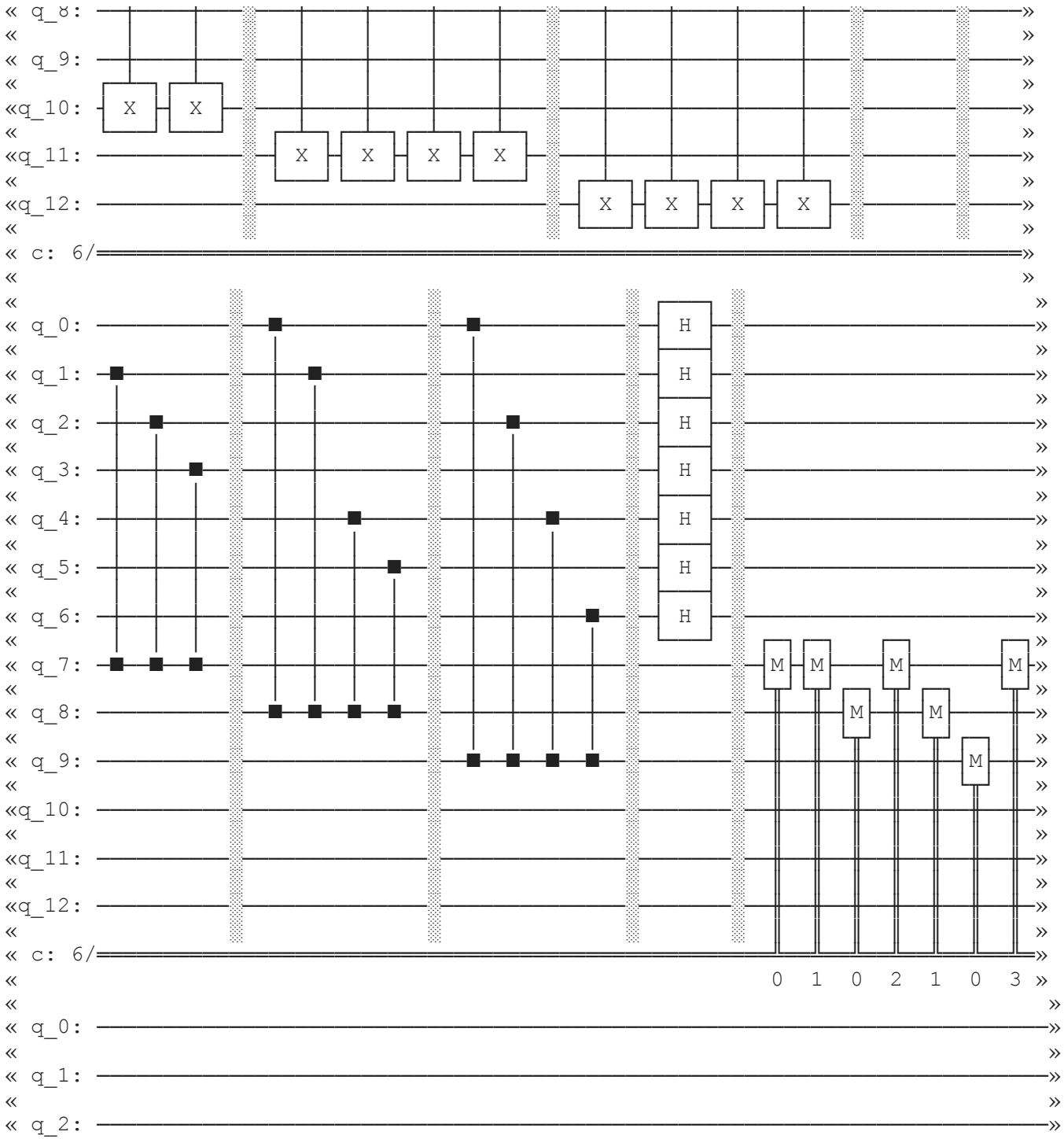


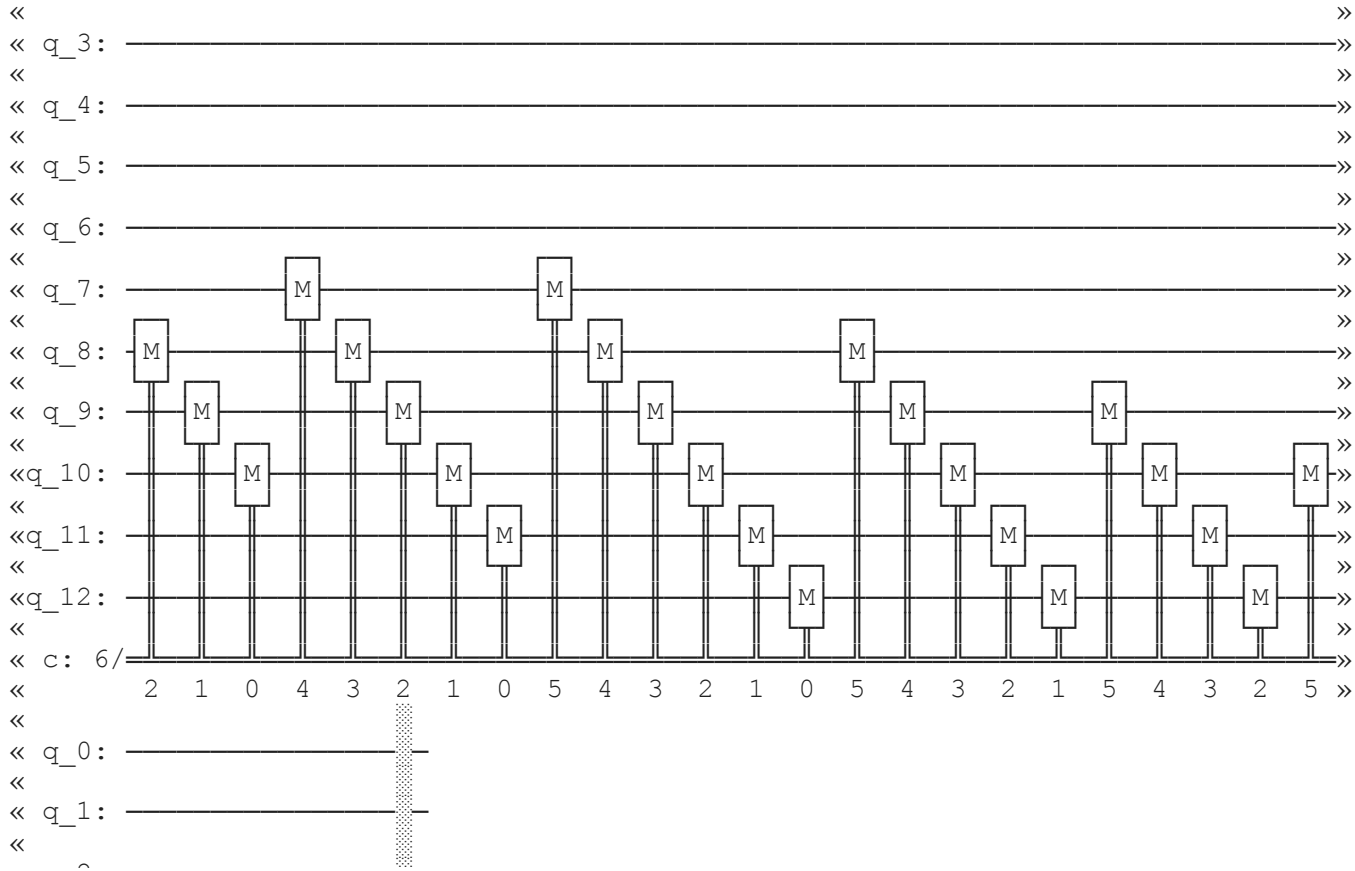
What is the error syndrome if we inject two bit-flip errors simultaneously, one on q0 and another on q1? Are these errors detectable and correctable? Why?

```
def error_injection(qc, d0, d1, d2, d3, d4, d5, d6):
  qc.x(d0)
```

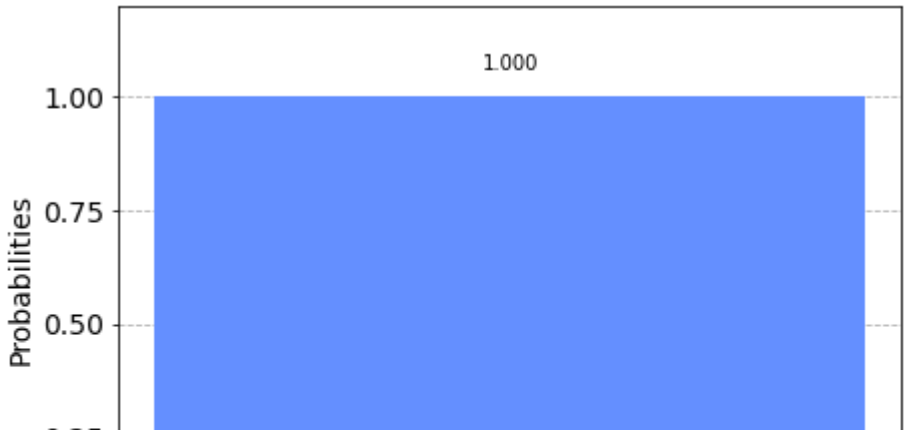
```
    qc.x(d1)

qr = QuantumRegister(13, name="q")
cr = ClassicalRegister(6, name="c")
## SETUP
steane_circuit = QuantumCircuit(qr,cr)

## STEP 1: encoding
encoding(steane_circuit,0,1,2,3,4,5,6)

## STEP 2: inserting errors
error_injection(steane_circuit,0,1,2,3,4,5,6)

## STEP 3:detection
detection(steane_circuit,0,1,2,3,4,5,6,7,8,9,10,11,12,cr)

steane_circuit.draw()
```


q_0:
q_1:
q_2:
q_3:
q_4:
q_5:
q_6:
q_7:
q_8:
q_9:
q_10:
q_11:
q_12:
c: 6/
H
X


q_8:
q_9:
q_10:
q_11:
q_12:
c: 6/
X
q_0:
q_1:
q_2:
q_3:
q_4:
q_5:
q_6:
q_7:
q_8:
q_9:
q_10:
q_11:
q_12:
c: 6/
H
M
0 1 0 2 1 0 3
q_0:
q_1:
q_2:




```
qasm_sim = Aer.get_backend('qasm_simulator')
t_qc = transpile(steane_circuit, qasm_sim)
qobj = assemble(t_qc)
results = qasm_sim.run(qobj).result()
counts = results.get_counts()
plot_histogram(counts)
```



NO ENTIENDO PORQUE ME DA ESTOS RESULTADOS EN LOS APARTADOS CUANDO HE COMPROBADO MIL VECES QUE EL CIRCUITO ESTUVIERA BIEN. NO ENTIENDO PORQUE CUANDO HAY UN ERROR CUALQUIERA TODO SE PONE A 1'S CUANDO HACEMOS EL HISTOGRAMA. HE ESTADO COMENTÁNDOLO CON ALGUNOS COMPAÑEROS Y EN LAS ÚLTIMAS DOS PREGUNTAS DEBERÍA HABER ERRORES Y SIN EMBARGO SE COMPORTA COMO LO HACE NORMALMENTE. POR ELLO NO HE PODIDO TERMINAR LA ÚLTIMA TABLA.

NO ENTIENDO PORQUE ME DA ESTOS RESULTADOS EN LOS APARTADOS CUANDO HE COMPROBADO MIL VECES QUE EL CIRCUITO ESTUVIERA BIEN. NO ENTIENDO PORQUE CUANDO HAY UN ERROR CUALQUIERA TODO SE PONE A 1'S CUANDO HACEMOS EL HISTOGRAMA. HE ESTADO COMENTÁNDOLO CON ALGUNOS COMPAÑEROS Y EN LAS ÚLTIMAS DOS PREGUNTAS DEBERÍA HABER ERRORES Y SIN EMBARGO SE COMPORTA COMO LO HACE NORMALMENTE. POR ELLO NO HE PODIDO TERMINAR LA ÚLTIMA TABLA.

✓ 0 s completado a las 19:49