

# QHack

## Quantum Coding Challenges

 CHALLENGE COMPLETED [View successful submissions](#)

**200 Superdense Coding** **200 points**

Zenaid and Reece have worked hard to implement classical computation reversibly. *This is all rather boring though*, says Trine. *"Let's introduce entanglement and have some fun!"* Trine tells them that the next [Laws of Infodynamics](#) will constrain how much classical information can be sent using entangled quantum resources.

3 3

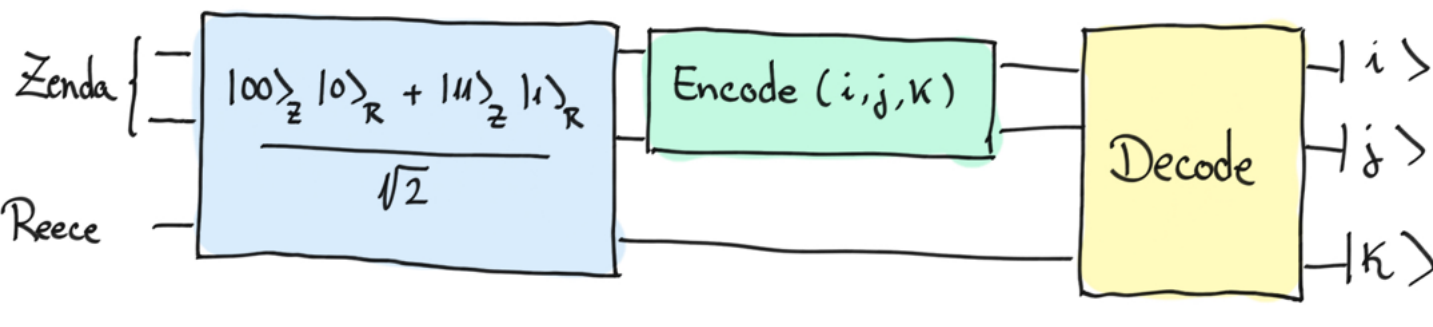
Superdense coding, for instance, uses Bell pairs to send two classical bits with a single qubit.

Time has inspired her usual Bell states, and instead provides Zelda with 2 qubits and Reece with 1 qubit. They are entangled forming the following state:

$$|\Psi\rangle = \frac{|00\rangle_Z|0\rangle_R + |11\rangle_Z|1\rangle_R}{\sqrt{2}}.$$

Zenda wants to send Reece 3 bits of information  $(i, j, k)$ , where each bit can take the value 0 or 1. To do so, she will apply an **encoding gate** on her two qubits and then send them to Bob.

Reece, who now has the three qubits and knows the strategy Zenda uses to encode, performs a decoding gate that will generate the state  $|i, j, k\rangle$ . Let's look at the following drawing to understand it.



Your goal will be to devise a coding and decoding strategy such that Reece can decode Zenda's bits, as shown in the figure above.

The strategy to build the encoding gate is akin to [superdense coding](#) for two bits of information, where all the Bell basis states are produced by acting only on one of the qubits in an entangled pair. Here, we want to produce all the GHZ basis states, which are analogous to the Bell basis for three qubits. They are given by

$$|GHZ\rangle_{ijk} = \frac{1}{\sqrt{2}}(|0jk\rangle + (-1)^i|1\bar{j}\bar{k}\rangle),$$

where  $\bar{j} = 1 - j$  and  $\bar{k} = 1 - k$ . However, we should do this by acting only on the first two qubits (i.e. Zenda's share) of the entangled state  $|\Psi\rangle$  that Trine provided. The decoding gate is also inspired by superdense coding, so reviewing the topic will help you a lot in this challenge!

## ▼ Laws of Infodynamics Part II: The Second and Third Laws

This box contains information that may be helpful, but is not essential to solving the problem. Suppose Zenda and Reece share a maximally entangled Bell pair of the form

$$|\Phi\rangle_{ZR} = \frac{|0\rangle_Z|0\rangle_R + |1\rangle_Z|1\rangle_R}{\sqrt{2}};$$

where the subscripts  $Z$  and  $R$  denote Zenda and Reece's share, respectively. This is one of four maximally entangled *Bell states*, labelled by two bits  $j$  and  $k$ :

$$|\beta(j, k)\rangle_{ZR} = \frac{1}{\sqrt{2}}(|0\rangle_Z |k\rangle_B R + (-1)^j |1\rangle_Z |k \oplus 1\rangle_R),$$

Half of an entangled state is called an entangled qubit or *ebit*. Since the ability to send an entangled qubit is a special

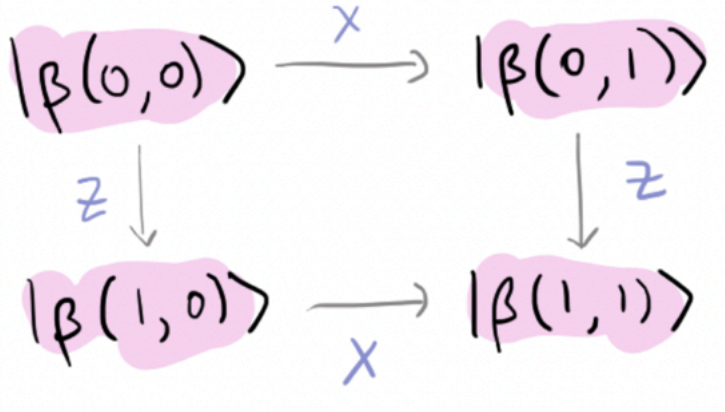
Half of an entangled state is called an entangled qubit or *ebit*. Since the ability to send an entangled qubit is a special case of sending a qubit, we have the Second Law of Infodynamics:

$$1 \text{ qubit} \geq 1 \text{ ebit}, \quad (2)$$

where  $x \geq y$  means having resource  $x$  also provides resource  $y$ . It's easy to check that

$$|\beta(j, k)\rangle = (Z^j X^k \otimes I)|\Phi\rangle.$$

We draw this below:



If Zenda and Reece share  $| \Phi \rangle$  (an ebit), and Zenda sends her qubit to Reece, Reece can measure in the Bell basis and learn  $j$  and  $k$ . Thus, an ebit and a qubit suffice to send two classical bits (or *cbits*). This protocol, called *superdense coding*, can be expressed as the Third Law of Infodynamics:

$$1 \text{ qubit} + 1 \text{ ebit} \geq 2 \text{ cbits.} \quad (3)$$

### Challenge code

You simply have to complete two quantum functions:

- `encode`: quantum function that will define an operator to be applied only on Zenda's qubits. This function will depend on the bits  $(i, j, k)$  to be encoded.
- `decode`: quantum function that defines the operator that Reece will use to retrieve the bits that Zenda sent. In this case, the operator does not have any information about Zenda's bits, so the same operators will always be applied regardless of the state that Zenda sends Reece.

## Output

In this challenge, we will not judge your solution using public or private test cases. Instead, we will check that, for all combinations of  $i$ ,  $j$ , and  $k$ , the entire encoding and decoding circuit behaves as expected.

Code ? Help

```

1 import pennylane as qml
2 import pennylane.numpy as np

3 def encode(i, j, k):
4     """
5     Quantum encoding function. It must act only on the first two qubits.
6     This function does not return anything, it simply applies gates.
7
8     Args:
9         i, j, k (int): The three encoding bits. They will take the values 1 or 0.
10    """

13 # Put your code here #

15 def decode():
16     """
17     Quantum decoding function. It can act on the three qubits.
18     This function does not return anything, it simply applies gates.
19    """

21 # Put your code here #

23 dev = qml.device("default.qubit", wires=3)
24
25 @qml.qnode(dev)
26 def circuit(i, j, k):
27     """
28     Circuit that generates the complete communication protocol.
29
30     Args:
31         i, j, k (int): The three encoding bits. They will take the value 1 or 0.
32    """
33
34     # We prepare the state 1/sqrt(2)(|000> + |111>)
35     qml.Hadamard(wires=0)
36     qml.CNOT(wires=[0, 1])
37     qml.CNOT(wires=[0, 2])
38
39     # Zenda encodes the bits
40     encode(i, j, k)
41
42     # Reece decode the information
43     decode()
44
45     return qml.probs(wires=range(3))

47 # These functions are responsible for testing the solution.
48
49 def run(test_case_input: str) -> str:
50
51     return None
52
53 def check(solution_output: str, expected_output: str) -> None:
54
55     for i in range(2):
56         for j in range(2):
57             for k in range(2):
58                 assert np.isclose(circuit(i, j, k)[4 * i + 2 * j + k], 1)
59
60         dev = qml.device("default.qubit", wires=3)
61
62         @qml.qnode(dev)
63         def circuit2(i, j, k):
64             encode(i, j, k)
65             return qml.probs(wires=range(3))
66
67         circuit2(i, j, k)
68         ops = circuit2.tape.operations
69
70         for op in ops:
71             assert not (2 in op.wires), "Invalid connection between qubits."

75 test_cases = [['No input', 'No output']]

76 for i, (input_, expected_output) in enumerate(test_cases):
77     print(f"Running test case {i} with input '{input_}'...")
78
79     try:
80         output = run(input_)
81
82     except Exception as exc:
83         print(f"Runtime Error. {exc}")
84
85     else:
86         if message := check(output, expected_output):
87             print(f"Wrong Answer. Have: '{output}'. Want: '{expected_output}'.")
88
89         else:
90             print("Correct!")

```