Challenge statement

The Tensor Tundra's environment is so cold, it's an excellent place to build quantum devices. A safe haven for those who venture into these icy plains is the Chalet of the Random Gate.

The front entrance of the chalet is a quantum gate, and its action will take us to different entry points: if it's an X gate, it takes us through the backdoor, if it's a Y gate, it takes us through the chimney, and if it's a Z gate, we just go through the front door.

This gate is a programmable device, so that we can choose where to end up (for some reason, people tend to avoid the Y gate). But when a strong blizzard strikes, the controls become frozen solid. Instead of being stuck in one setting, the noise coming from the blizzard makes the gate act randomly.

In this challenge, we explore what happens when the application of a quantum gate is not done deterministically. Suppose that we have a single qubit $|0\rangle$ on which a single gate acts at random. This gate can be

- The Pauli X gate, acting with probability p,
- The Pauli Y gate, acting with probability q,
- The Pauli Z gate, acting with probability r,
- The Identity I, acting with probability $1 - p - q - r$.

You are asked to calculate the measurement probabilities in the computational basis upon the application of these random gates.

Fingers crossed that you don't get stuck in the chimney!

Challenge Code

In the challenge template, you must complete the `random_gate` QNode which, given the probabilities `p`, `q`, and `r` (`float`) return the measurement probabilities (`np.array(float)`) in the computational basis.

Input

As an input to this challenge, you are given the probabilities p, q and r (`float`) representing the application probabilities of the Paul X, Y, and Z gates respectively.

Output

The output is an `np.tensor` $[p_0, p_1]$ containing the probabilities of measuring 0 or 1 in the computational basis after the random application of the gate.

Test cases

The following **public test cases** are available for you to check your work. There are also some **hidden test cases** which we will use to check that your solution works in full generality.

```
test_input: [0.25,0.25,0.25]
expected_ouput: [0.5,0.5]

test_input: [0.125,0.25,0.2]
expected_ouput: [0.625, 0.375]
```

If your solution matches the correct one up to a relative tolerance of 1×10^{-4}, the output will be `"Success!"`. Otherwise, you will receive an `"Incorrect"` prompt.

Good luck!