Goal: Write a function to turn a lock up or down a given number of times.

Previously, you used parameters to define a `move` function with an input, `distance`. In this puzzle, you'll define a `turnLock` function that uses the parameters `up` and `numberOfTimes` to determine the direction and number of times your expert should turn the lock.

> turnLock parameters explained
>
> `up` takes an input of type Bool (Boolean), indicating whether to turn the lock **up** (`true`) or **down** (`false`).
> `numberOfTimes` takes an input of type Int, indicating the number of times to turn the lock.

1. Use both parameters, `up` and `numberOfTimes`, to define your function.
2. Check the value of `up` to determine if you should call `turnLockUp()` or `turnLockDown()`.
3. Use the `numberOfTimes` value to determine how many times to run either `turnLockUp()` or `turnLockDown()`.

```
let expert = Expert()
let character = Character()

func turnLock(up: Bool, numberOfTimes: Int) {
    for i in 1...numberOfTimes {
        if up == true {
            expert.turnLockUp()
        } else {
            expert.turnLockDown()
        }
    }
}
func expertTurnAround() {
    expert.turnLeft()
    expert.turnLeft()
}

func characterTurnAround() {
    character.turnLeft()
    character.turnLeft()
}

expertTurnAround()
turnLock(up: true, numberOfTimes: 3)
character.moveForward()
character.turnRight()
character.moveForward()
character.turnRight()
turnLock(up: false, numberOfTimes: 3)
character.moveForward()
character.collectGem()
characterTurnAround()
character.moveForward()
character.moveForward()
expertTurnAround()
turnLock(up: true, numberOfTimes: 1)
character.moveForward()
character.collectGem()
characterTurnAround()
character.moveForward()
character.turnRight()
character.moveForward()
character.turnRight()
turnLock(up: true, numberOfTimes: 2)
character.moveForward()
character.collectGem()
```