Goal: Use the Right-Hand Rule algorithm to navigate around walls.

Run this puzzle, and notice how your character stops after the first gem. The algorithm used here follows the right-hand rule to move around walls. To solve the puzzle, you'll need to tweak the algorithm, but first try using pseudocode to plan the action.

Pseudocode looks a bit like Swift code, but it's worded and structured so humans can easily understand it.

Example

```
navigate around wall {
   if blocked to the right {
      move forward
   } else {
      turn right
      move forward
   }
}

while not on closed switch {
   navigate around wall
   if on gem {
      collect gem
      turn around
   }
}
toggle switch
```

1 Based on the pseudocode above, write out a solution in code for the puzzle.

2 Run your code and tweak your algorithm, if necessary, to solve the puzzle.

```
func navigateAroundWall() {
    if isBlockedRight {
        moveForward()
    }
    else {
        turnRight()
        moveForward()
    }
    if isOnGem {
        turnLeft()
        turnLeft()
        collectGem()
```

```
    }
    if isOnClosedSwitch {
        toggleSwitch()
    }
}
    while !isOnGem && !isOnClosedSwitch {
        // Quote Wrapped expressed all 3 HAS to be true to "break" out
         of loop
        if (isBlocked && isBlockedLeft && isBlockedRight)
        {
            break
        }
        else {
            navigateAroundWall()
        }
        }
```