

# Recursion Quiz

## ANSWER KEY

Write code to recursively solve the following problem:

You want to know if an integer  $n$  is an odd number. Normally you'd just see `if (n%2 == 0)` but the `%` key on your keyboard is broken. Code up an `isOdd` function recursively without using the `%` mod function. Here is a math description of the function.

$$isOdd(x) = \begin{cases} no & \text{if } x == 0 \\ yes & \text{if } x == 1 \\ isOdd(x-2) & \text{otherwise} \end{cases}$$

Comments and visibility keywords not necessary.

What happens if someone tries this on negative numbers? Can you fix your method to handle negative numbers?

Original function implementation:

```
boolean isOdd(int x) {
    if (x == 0)
        return false;
    else if (x == 1)
        return true;
    else
        return isOdd(x-2);
}
```

If someone tried to determine the parity of a negative number using this implementation, they would (possibly) never find a solution as it could decrement the number forever. Alternately, the decrement may eventually cause the integer to overflow and start again from the maximum value. There is no guarantee that the system is implemented to properly handle this overflow. As a result, the value returned may not accurately represent the parity of the negative number.

To fix this situation, you can simply take the absolute value of the negative number.

The fixed implementation:

```
boolean isOdd(int x) {
    if (x < 0) isOdd(-x);
    if (x == 0) return false;
    else if (x == 1) return true;
    else return isOdd(x-2);
}
```