1

2

3

4

5

6

Number Game

Alex has an integer and wants to convert it to *0* using the following operations on its binary representation:

- Change the i^{th} binary digit only if $(i+1)^{th}$ binary digit is 1 and all other binary digits from *(i+2)* to the end are zeros.
- Change the rightmost digit without restriction.

Alex can use the above operations as many times as necessary, but wants to determine the minimum number of operations required. For example, given the number $n = 8_{10} = 1000_2$. *15* operations are required to convert the number to zero under the rules:

```
1000→1001→1011→1010→1110→1111→1101→1100→0100→0101→0111→0110→0010→0011→0001→0000
```

Note: In the binary representation of a number, the binary digit's positions are numbered as *0* to *x* from left to right, where *x* is the number of digits in the binary representation of the number.

Function Description
Complete the function *minOperations* in the editor below. The function must return an integer that denotes the minimum number of operations required to covert *n* to *0.*

minOperations has the following parameter(s):
n: integer, the number Alex has.

Constraints

- $1 \le n \le 10^{15}$

Input Format For Custom Testing

Sample Case 0

Sample Input For Custom Testing

```
13
```

Sample Output

```
9
```

Explanation
The binary representation of 13 is 1101. This is the sequence of steps that change 13 to 0 in this game.

```
1101→1100→0100→0101→0111→0110→0010→0011→0001→0000
```

Sample Case 1

YOUR ANSWER

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour.
Start tour

For help on how to read input and write output in Python 3, click here.

Draft saved 11:30 am   Original Code   Python 3

```
#!/bin/python3…

DEBUG = False
DEBUG_FUNCTIONS = False
MEMOIZE = True
```

1

2

3

4

5

6

```
    indent = 0
    active = DEBUG_FUNCTIONS

    def indent_inc(self):
        self.indent += 1

    def indent_dec(self):
        self.indent -= 1

    def log(self, m):
        if self.active:
            print(" "*self.indent + m)

logger = Logger()

# ---------------------------------------------------------------------------- #

def debugDecorator():
```

Line: 170 Col: 13

☐ **Test against custom input**

Run Code | Submit code & Continue

(You can submit any number of times)

Download sample test cases *The input/output files have Unix line endings. Do not use Notepad to edit them on windows.*

Compiled successfully. All available test cases passed!

Tip: Debug your code against custom input

Test Case #1:	✓	Test Case #4:	✓ Success	Test Case #7:	✓
Test Case #2:	✓	Test Case #5:	✓	Test Case #8:	✓
Test Case #3:	✓	Test Case #6:	✓		

Testcase 1: *Success*

Input [Download]

```
13
```

Your Output

```
9
```

Expected Output [Download]

```
9
```

Testcase 2: *Success*

Input [Download]

```
11
```

Your Output

```
13
```

Expected Output [Download]

```
13
```

1

2

3

4

5

6

```
156
```

Your Output

```
232
```

Expected Output [Download]

```
232
```

Testcase 4: *Success*

Your Output

```
Output hidden
```

Testcase 5: *Success*

Your Output

```
Output hidden
```

Testcase 6: *Success*

Your Output

```
Output hidden
```

Testcase 7: *Success*

Your Output

```
Output hidden
```

Testcase 8: *Success*

Your Output

```
Output hidden
```