Decimal to binary

Given an integer N, convert to its binary representation.

Idea: As long as the given number is greater than zero

① create a placeholder for the answer.

② start a loop and at each iteration.
add the result of (n%2) to result.
update n to n//2.

③ when n<=0, break out of the loop.

```
def toBinary(N):
    answer=" "
    while N>0:
        removed=(N%2)
        // Add the remainder to the answer
        answer = answer + str(removed)
        // update N by removing its last digit
        N = N//2
    return answer
```

lets say N=7, we want to find its binary representation

N	N%2	answer	N//2	N>0
7	1	1	7//2=3	yes
3	3%2 = 1	"1"+"1"="11"	3//2=1	Yes
1	1%2 = 1	"11" +"1"= "111"	1//2=0	No, break, return answer

Now, if the goal is to find the representation for all numbers in range of 1 to N, The above function should be called for each integer.

```
def generate(n):
        answer = []
        for i in range(1, n+1):
                // calling toBinary function for each number and saving the result
                answer.append(toBinary(i))
        return answer
```

Analysis

def toBinary(N): It can be observed that The growth of This function is depends on integer N and at every Iteration we are reducing by half and we stop when N becomes 1

$N=10$	$N=16$	$\log_2(16) = \log_2 2^4 = 4$
$10//2 = 5$	$16//2 = 8$	
$5//2 = 2$	$8//2 = 4$	$\log_2(10) = \log_2 2^{3.\cdots} = 3$
$2//2 = 1$ stop	$4//2 = 2$	
	$2//2 = 1$ stop	

Conclusion: It can observed the above function is a logarithmic function (Runs $\log_2 N$ times) def generate(n): it has a simple for loop and that Runs N times, but inside the loop there is another functio which does $\log_2 N$ operations.

T·C = time complexity

overall T·C = outer loop * inner loop

$= N * \log_2 N$

Note: same goes with space complexity.