Count numbers that don’t contain 3

Given a number n, write a function that returns count of numbers from 1 to n that don’t contain digit 3 in their decimal representation.

Examples:

```
Input: n = 10
Output: 9

Input: n = 45
Output: 31
// Numbers 3, 13, 23, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 43 contain digit 3.

Input: n = 578
Ouput: 385
```

Solution:
We can solve it recursively. Let count(n) be the function that counts such numbers.

```
'msd' --> the most significant digit in n
'd'   --> number of digits in n.

count(n) = n if n < 3

count(n) = n - 1 if 3 <= n < 10

count(n) = count(msd) * count(10^(d-1) - 1) +
           count(msd) +
           count(n % (10^(d-1)))
           if n > 10 and msd is not 3

count(n) = count( msd * (10^(d-1)) - 1)
           if n > 10 and msd is 3
```

```
Let us understand the solution with n = 578.
count(578) = 4*count(99) + 4 + count(78)
The middle term 4 is added to include numbers 100, 200, 400 and 500.

Let us take n = 35 as another example.
count(35) = count (3*10 - 1) = count(29)
```

```
#include <stdio.h>

/* returns count of numbers which are in range from 1 to
```

```
   as a digit */
int count(int n)
{
    // Base cases (Assuming n is not negative)
    if (n < 3)
        return n;
    if (n >= 3 && n < 10)
       return n-1;

    // Calculate 10^(d-1) (10 raise to the power d-1) wh
    // number of digits in n. po will be 100 for n = 578
    int po = 1;
    while (n/po > 9)
        po = po*10;

    // find the most significant digit (msd is 5 for 578
    int msd = n/po;

    if (msd != 3)
      // For 578, total will be 4*count(10^2 - 1) + 4 +
      return count(msd)*count(po - 1) + count(msd) + cou
    else
      // For 35, total will be equal to count(29)
      return count(msd*po - 1);
}

// Driver program to test above function
int main()
{
    printf ("%d ", count(578));
    return 0;
}
```

Output:

```
385
```