# Remove 9

Start from integer 1, remove any integer that contains 9 such as 9, 19, 29...

So now, you will have a new integer sequence: 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, ...

Given a positive integer `n`, you need to return the n-th integer after removing. Note that 1 will be the first integer.

### Example 1:

```
Input: 9
Output: 10
```

**Hint**: n will not exceed `9 x 10^8`.

## Solution 1

This is a radix problem.
Just change decimal to 9-based.

```
public int newInteger(int n) {
    return Integer.parseInt(Integer.toString(n, 9));
}
```

Of course, you can write it yourself.

```
public int newInteger(int n) {
 int ans = 0;
 int base = 1;

 while (n > 0){
  ans += n % 9 * base;
  n /= 9;
  base *= 10;
 }
 return ans;
}
```

written by DemonSong original link here

## Solution 2

The set of numbers without 9s is the same as the set of base-9 numbers, and they occur in the same order. The answer is therefore just the n-th base-9 number.

```
def newInteger(self, n):
    ans = ''
    while n:
        ans = str(n%9) + ans
        n /= 9
    return int(ans)
```

written by awice original link here

## Solution 3

```
int newInteger(int n) {
    int res = 0, s = 1;
    while (n > 0) {
        res += n % 9 * s;
        n /= 9;
        s *= 10;
    }
    return res;
}
```

You may already know binary and decimal where number goes up to a more significant digit when it hits two or ten. e.g. 2^1=>10, 2^2=>100, 2^3=>1000. The problem here is almost the same. When it hits 9 it goes up. e.g. 9^1=>10, 9^2=>100, 9^3=>1000, etc. Simply use our math knowledge to convert back making use of division and modulo.

written by mzchen original link here