**Problem :** [**https://codingcompetitions.withgoogle.com/kickstart/round/000000000019ff49/000000000043b0c6#problem**](https://codingcompetitions.withgoogle.com/kickstart/round/000000000019ff49/000000000043b0c6#problem)

**Approach :**

-> There are 4 things here that decide the STATE :

**{position,numberStarted,parity,bounded}**

-> At the 1st position, we can either start the number by placing odd numbers,and the next place **parity will be even,numberStarted=1,bounded will depend on what digit we placed.**

(OR) we can put a 0 and not start the number. So that way we will collect the amount of numbers wich have n-1 digits,n-2 and so on recursively.

So for the next position,

**parity will be odd** as number still not started,

**number started = 0**(so next number can decide to either start or not start,

**bounded=0(imp)**(bcoz no way now using less digits we can form a number greater than original number given)

-> For example for a 3 digit number we decide to place a 0 at 1st position , then we will collect all 2 digit nos and 1 digit nos , HOW?

The 2nd place will be called recursively, if it decides to place a 0 there,then all 1 digit numbers will be recursively collected, and if it decides to start the number all 2 digit numbers will be collected,

**At the 1st place ,   
if number started = 0,then now, we have to compulsorily start the number,**so 5 nos 1,3,5,7,9 will satisfy as its not bounded , and 5 will be returned.

**if number started = 1,then what numbers to put will depend on parity and whether we are bounded or not.**

**Format : {position,numberStarted,parity,bounded},assume 4 digit number**

=> So in short we will call initially dp[1][0][1][1] from main function :

So if we decide to **not start** the number then dp[2][0][1][0] will be called and it will return

dp[2][0][1][0] (means not started the number to collect 2 and 1 digit numbers) + dp[2][1][1][0](started the number to collect 3 digit numbers)

And if number started we will call dp[2][1][0][bounded depends] and we will collect all 4 digit numbers.

Ans would definitely be addition of both.

-> **One thing to note is if not bounded, then always 5 numbers will satisfy irrespective of current parity(0,2,4,6,8 or 1,3,5,7,9).**

Track the above process by taking a small example of a 2 digit number and then its easy to understand.

**Code :** [**https://ideone.com/m7QPH8**](https://ideone.com/m7QPH8)