**Q. Number Inversions**

Rahul is facing a unique problem, which he doesn't know how to solve. The problem asks him to build the smallest possible number by applying inversion on any digit of the number any number of times.

Inversion of a digit is defined as that digit being replaced by 9 minus that digit. Meaning that inversion of 9 will be 9 - 9 = 0 and inversion of 1 will be 9 - 1 = 8 and so on.

The final output should not have any leading zeroes.

Input Format:

The only line of the input contains an integer N

Output Format:

Print only one single integer on a line as described above.

Constraints:

1 <= N <= 10^18

Examples:

Input:

87

Output:

12

Explanation:

9-8 = 1 and 9 -7 = 2; if we see carefully, 12 is the smallest possible number we can get.

Example:

Input:

99

Output:

90

Explanation:

In this case, we do not replace the first 9 since it will lead to a leading zero.

**Python Program**

N = input()  
global counter1  
counter1 = 0  
  
def inversion(new\_N):  
 for counter in range(1):  
 new\_N = str(new\_N)[:counter1] + str(9 - int(str(new\_N)[counter1])) +str(new\_N)[(counter1+1):]  
 return new\_N  
  
  
if 1 <= int(N) <= (10\*\*18):  
 Flag = True  
 new\_number = N  
 while Flag:  
 small\_number = inversion(new\_number)  
 if small\_number[0] == **'0'**:  
 counter1 += 1  
 if (counter1+1) > len(str(N)):  
 print(new\_number)  
 break  
 elif int(small\_number) <= int(new\_number):  
 new\_number = small\_number  
 else:  
 counter1 += 1  
 if (counter1+1) > len(str(N)):  
 print(new\_number)  
 break