You are given a non-negative integer number. Your task is to find the closest prime number to it, and calculate the distance between them.

**Example**

* For number = 0, the output should be  
  primeDistance(number) = 2.

The closest prime number to 0 is 2, so the answer is 2 - 0 = 2.

* For number = 11, the output should be  
  primeDistance(number) = 0.

11 is a prime number, so the answer is 11 - 11 = 0.

**Input/Output**

* **[time limit] 4000ms (py)**
* **[input] string number**

*Constraints:*  
0 ≤ int(number) ≤ 9 · 1014.

* **[output] integer**

<https://codefights.com/challenge/e68HxZ8Qo5jF7Mjxi>

import math

def **\_try\_composite**(a, d, n, s):

if pow(a, d, n) == 1:

return False

for i in range(s):

if pow(a, 2\*\*i \* d, n) == n-1:

return False

return True # n is definitely composite

<https://rosettacode.org/wiki/Miller%E2%80%93Rabin_primality_test#Java>

def **is\_prime**(n, \_precision\_for\_huge\_n=16):

if n in \_known\_primes or n in (0, 1):

return True

if any((n % p) == 0 for p in \_known\_primes):

return False

d, s = n - 1, 0

while not d % 2:

d, s = d >> 1, s + 1

# Returns exact according to http://primes.utm.edu/prove/prove2\_3.html

if n < 1373653:

return not any(\_try\_composite(a, d, n, s) for a in (2, 3))

if n < 25326001:

return not any(\_try\_composite(a, d, n, s) for a in (2, 3, 5))

if n < 118670087467:

if n == 3215031751:

return False

return not any(\_try\_composite(a, d, n, s) for a in (2, 3, 5, 7))

if n < 2152302898747:

return not any(\_try\_composite(a, d, n, s) for a in (2, 3, 5, 7, 11))

if n < 3474749660383:

return not any(\_try\_composite(a, d, n, s) for a in (2, 3, 5, 7, 11, 13))

if n < 341550071728321:

return not any(\_try\_composite(a, d, n, s) for a in (2, 3, 5, 7, 11, 13, 17))

# otherwise

return not any(\_try\_composite(a, d, n, s)

for a in \_known\_primes[:\_precision\_for\_huge\_n])

\_known\_primes = [2, 3]

\_known\_primes += [x for x in range(5, 1000, 2) if is\_prime(x)]

#print is\_prime(31, \_precision\_for\_huge\_n=16)

def **primeDistance**(number):

n = int(number)

if n == 0: return 2

if n == 1: return 1

if is\_prime(n):

return 0

r = n

l = n

if n!=0 and n % 2 ==0:

r+=1

l-=1

while not is\_prime(r):

r+=2

while l >= 0 and not is\_prime(l):

l-=2

print l

print r

if l < 0:

return r-n

return min(r-n, n - l)