Python-阿里巴巴找黄金宝箱(III)-一贫如洗的樵夫阿里巴巴在去砍柴的路上

题目描述：

一贫如洗的樵夫阿里巴巴在去砍柴的路上，无意中发现了强盗集团的藏宝地，藏宝地有编号从0~N 的箱子，每个箱子上面贴有一个数字。

阿里巴巴念出一个咒语数字，查看宝箱是否存在两个不同箱子，这两个箱子上贴的数字相同，同时这两个箱子的编号之差的绝对值小于等于咒语数字，

如果存在这样的一对宝箱，请返回最先找到的那对宝箱左边箱子的编号，如果不存在则返回-1。

输入描述：

第一行输入一个数字字串，数字之间使用逗号分隔，例如: 1,2,3,1

字串中数字个数>=1，<=100000；每个数字值>=-100000，<=100000；

第二行输入咒语数字，例如：3，咒语数字>=1，<=100000

输出描述：

存在这样的一对宝箱，请返回最先找到的那对宝箱左边箱子的编号，如果不存在则返回-1

补充说明：

示例1
输入: 6,3,1,6
3
输出: 0
说明:

示例2
输入: 5,6,7,5,6,7
2
输出: -1
说明:

```
from io import *
import os, sys
from math import *
from collections import defaultdict

BUFSIZE = 4096


class FastIO(IOBase):
    newlines = 0

    def __init__(self, file):
        self._fd = file.fileno()
```

```
            self.buffer = BytesIO()
            self.writable = "x" in file.mode or "r" not in file.mode
            self.write = self.buffer.write if self.writable else None

        def read(self):
            while True:
                b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
                if not b:
                    break
                ptr = self.buffer.tell()
                self.buffer.seek(0, 2), self.buffer.write(b), self.buffer.seek(ptr)
            self.newlines = 0
            return self.buffer.read()

        def readline(self):
            while self.newlines == 0:
                b = os.read(self._fd, max(os.fstat(self._fd).st_size, BUFSIZE))
                self.newlines = b.count(b"\n") + (not b)
                ptr = self.buffer.tell()
                self.buffer.seek(0, 2), self.buffer.write(b), self.buffer.seek(ptr)
            self.newlines -= 1
            return self.buffer.readline()

        def flush(self):
            if self.writable:
                os.write(self._fd, self.buffer.getvalue())
                self.buffer.truncate(0), self.buffer.seek(0)


    class IOWrapper(IOBase):
        def __init__(self, file):
            self.buffer = FastIO(file)
            self.flush = self.buffer.flush
            self.writable = self.buffer.writable
            self.write = lambda s: self.buffer.write(s.encode("ascii"))
            self.read = lambda: self.buffer.read().decode("ascii")
            self.readline = lambda: self.buffer.readline().decode("ascii")


    sys.stdin = IOWrapper(sys.stdin)
    sys.stdout = IOWrapper(sys.stdout)
    input = lambda: sys.stdin.readline().rstrip("\r\n")
```

```
def I():
    return input()


def II():
    return int(input())


def MII():
    return map(int, input().split())


def LI():
    return list(input().split())


def LII():
    return list(map(int, input().split()))


def GMI():
    return map(lambda x: int(x) - 1, input().split())


def LGMI():
    return list(map(lambda x: int(x) - 1, input().split()))


def solve():
    s = I()
    c = II()    # curse
    d = defaultdict(lambda: -1)
    l = s.split(",")
    for i, x in enumerate(l):
        if d[x] != -1 and i - d[x] <= c:
            print(d[x])
            return
        d[x] = i
    print(-1)


if __name__ == "__main__":
    t = 1
    while t > 0:
```

```
solve()
t -= 1
```