寻找数组的中心下标

```
#!/usr/bin/python
# -*- coding:UTF-8 -*-
# Filename: for.py

'''
给你一个整数数组 `nums` ，请计算数组的 **中心下标** 。

数组 **中心下标** 是数组的一个下标，其左侧所有元素相加的和等于右侧所有元素相加的和。

如果中心下标位于数组最左端，那么左侧数之和视为 `0` ，因为在下标的左侧不存在元素。这一点对于中心下标位于数组最右端同样适用。

如果数组有多个中心下标，应该返回 **最靠近左边** 的那一个。如果数组不存在中心下标，返回 `-1`
'''

# 预处理  nums = [1, 2, 3]
lst = input()
true_lst = lst[8:-1].split(', ')
true_lst = [int(x) for x in true_lst]
flag = 0
# 循环检测
for i in range (len(true_lst)):
    if(i):
        left = sum(true_lst[:i])
        right = sum(true_lst[i+1:])
        if(left == right):
            print(i)
            flag = 1
            break
    else:
        right = sum(true_lst[i+1:])
        if(right == 0):
            print(i)
            flag = 1
            break
if(flag == 0):
    print(-1)
```

```
  SyntaxError: invalid syntax
● (base) [root@cpy /]# /home/app/anaconda3/bin/python /home/prj/tes1.py
  nums = [1, 7, 3, 6, 5, 6]
  3
● (base) [root@cpy /]# /home/app/anaconda3/bin/python /home/prj/tes1.py
  nums = [1, 2, 3]
  -1
● (base) [root@cpy /]# /home/app/anaconda3/bin/python /home/prj/tes1.py
  nums = [2, 1, -1]
  0
○ (base) [root@cpy /]#
```

计算数组中心位置

```python
#!/usr/bin/python
# -*- coding:UTF-8 -*-
# Filename: test.py

'''
给你一个整数数组 nums，请计算数组的中心位置，数组的中心位置是数组的一个下标，

**其左侧所有元素相乘的积等于右侧所有元素相乘的积。数组第一个元素的左侧积为 1，最后一个元素的右侧积为 1。**

如果数组有多个中心位置，应该返回最靠近左边的那一个，如果数组不存在中心位置，返回**-1**。
'''

lst = input().split(' ')
# true_lst = lst[7:]
true_lst = [int(x) for x in lst]

flag = 0

for i in range (len(true_lst)):
    left = right = 1
    for j in range (i):
        left *= true_lst[j]
    for k in range (i+1, len(true_lst)):
        right *= true_lst[k]
    if left == right:
        print(i)
        flag = 1
        break
if flag == 0:
```

```
    print(-1)
```

```
KeyboardInterrupt
(base) [root@cpy /]# /home/app/anaconda3/bin/python /home/prj/tes2.py
2 5 3 6 5 6
3
(base) [root@cpy /]#
```

模拟打折

```
#!/usr/bin/python
# -*- coding:UTF-8 -*-
# Filename: test.py

import math
'''
模拟商场优惠打折，有三种优惠券可以用，**满减券**、**打折券**和**无门槛券**。

**满减券：满 100 减 10，满 200 减 20，满 300 减 30，满 400 减 40，以此类推不限制使用**；

**打折券：固定折扣 92 折，且打折之后向下取整，每次购物只能用 1 次**；

**无门槛券：一张券减 5 元，没有使用限制**。

每个人结账使用优惠券时有以下限制：

每人每次只能用两种优惠券，并且同一种优惠券必须一次用完，不能跟别的穿插使用（比如用一张满减，再用一张打折，再用一张满减，这种顺序不行）。

求不同使用顺序下每个人用完券之后得到的最低价格和对应使用优惠券的总数；如果两种顺序得到的价格一样低，就取使用优惠券数量较少的那个。


输入：第一行三个数字 m,n,k，分别表示每个人可以使用的满减券、打折券和无门槛券的数量。
3 2 5
3
100
200
400

输出：
65 6
135 8
```

```
275 8
'''

m, n, k = map(int, input().split())
person = int(input())
price = []
# num = []
for i in range(person):
    price.append(int(input()))

def manjian(m, price, num):
    while((price >= 400) & (m >= 1)):
        price -= 40
        m -= 1
        num += 1
    while((price >= 300) & (m >= 1)):
        price -= 30
        m -= 1
        num += 1
    while((price >= 200) & (m >= 1)):
        price -= 20
        m -= 1
        num += 1
    while((price >= 100) & (m >= 1)):
        price -= 10
        m -= 1
        num += 1
    return price, num

def dazhe(price):
    price = price * 0.92
    # 向下取整
    return int(price)


def wumenkan(k, price):
    while(k >= 1):
        price -= 5
        k -= 1
    return price

# 满减券 + 打折券
def func1(m, n, k, price, num):
    # 先用满减券
```

```
    p1 = p2 = price
    p1, num1 = manjian(m, p1, num)
    p1 = dazhe(p1)
    num1 += 1
    # print("man + dazhe", p1)
    # 先用打折券
    p2 = dazhe(p2)
    p2, num2 = manjian(m, p2, num)
    num2 += 1
    # print("dazhe + man", p2)
    if(p1 < p2):
        return p1, num1
    else:
        return p2, num2
    # return min(p1, p2), num
# 满减券 + 无门槛券
def func2(m, n, k, price, num):
    p1 = p2 = price
    p1, num1 = manjian(m, p1, num)
    p1 = wumenkan(k, p1)
    num1 += k
    # print("man + wu", p1)
    p2 = wumenkan(k, p2)
    num2 = 0
    num2 += k
    p2, num2 = manjian(m, p2, num)
    # print("wu + man", p2)
    if(p1 < p2):
        return p1, num1
    else:
        return p2, num2
# 打折券 + 无门槛券
def func3(m, n, k, price, num):
    p1 = p2 = price
    p1 = dazhe(p1)
    num1 = 0
    num1 += 1
    p1 = wumenkan(k, p1)
    num1 += k
    # print("dazhe + wu", p1)
    p2 = wumenkan(k, p2)
    num2 = 0
    num2 += k
    p2 = dazhe(p2)
```

```
    num2 += 1
    # print("wu + dazhe", p2)
    if(p1 < p2):
        return p1, num1
    else:
        return p2, num2
# 满减券 + 打折券 + 无门槛券
def func4(m, n, k, price, num):
    p1, num1 = func1(m, n, k, price, num)
    p2, num2 = func2(m, n, k, price, num)
    p3, num3 = func3(m, n, k, price, num)
    if((p1 < p2) & (p1 < p3)):
        return p1, num1
    elif((p2 < p1) & (p2 < p3)):
        return p2, num2
    else:
        return p3, num3
    # return min(p1, p2, p3), num

for i in range(person):
    min_p, num = func4(m, n, k, price[i], 0)
    print(min_p, num)
```

```
(base) [root@cpy /]# /home/app/anaconda3/bin/python /home/prj/tes5.py
3 2 5
3
100
200
400
65 6
135 8
275 8
(base) [root@cpy /]#
```

预定酒店

```
#!/usr/bin/python
# -*- coding:UTF-8 -*-
# Filename: test.py

'''
放暑假了，小明决定到某旅游景点游玩，他在网上搜索到了各种价位的酒店（长度为 n 的数组 A），
```

```
他的心理价位是 x 元，请帮他筛选出 k 个最接近 x 元的酒店（n>=k>0）,并**由低到高**打印酒店的价格。

**输入描述**

第一行：n, k, x
第二行：A[0] A[1] A[2]…A[n-1]
'''

n, k, x = map(int, input().split())
prices = sorted(list(map(int, input().split())))

# 所有数组元素减去x，取绝对值为key，价格为value
diff = {}
for i in range(len(prices)):
    # 如果差值的绝对值已经存在，那么将价格加入到列表中
    if abs(prices[i] - x) in diff:
        diff[abs(prices[i] - x)].append(prices[i])
    else:
        diff[abs(prices[i] - x)] = [prices[i]]
# 按照key排序
diff = sorted(diff.items(), key=lambda x: x[0])
ans = []
for i in range(len(diff)):
    for j in diff[i][1]:
        ans.append(j)
        if len(ans) == k:
            break
ans = sorted(ans[:k])
print(ans)
```

```
(base) [root@cpy /]# /home/app/anaconda3/bin/python /home/prj/tes3.py
10 5 6
1 2 3 4 5 6 7 8 9 10
[4, 5, 6, 7, 8]
(base) [root@cpy /]#
```

寻找相似单词

```
#!/usr/bin/python
# -*- coding:UTF-8 -*-
# Filename: test.py
```

```
'''
   给定一个可存储若干单词的字典，找出指定单词的所有相似单词，并且按照单词名称从小到大排序输出。

   单词仅包括字母，但可能大小写并存（大写不一定只出现在首字母）。

   相似单词说明：给定一个单词 X，如果通过任意交换单词中字母的位置得到不同的单词 Y，那么定义 Y 是 X 的相似单词，如 abc、bca 即为相似单词（大小写是不同的字母，如 a 和 A 算两个不同字母）。

   字典序排序： 大写字母<小写字母。同样大小写的字母，遵循 26 字母顺序大小关系。

   即*A<B<C<…<X<Y<Z<a<b<c<…<x<y<z*. 如*Bac<aBc<acB<cBa*.

**输入描述：**

   第一行为给定的单词个数 N（N 为非负整数）

   从第二行到地 N+1 行是具体的单词（每行一个单词）

   最后一行是指定的待检测单词（用于检测上面给定的单词中哪些是与该指定单词是相似单词，该单词可以不是上面给定的单词）

4
abc
dasd
tad
bca
abc

----->   abc bca
'''
n = int(input())
words = []
for i in range(n):
    words.append(input())
target = input()

# initial dict for words
# 所有字母包括大小写都为0
dic_init = {}
for i in range(26):
    dic_init[chr(ord('a')+i)] = 0
    dic_init[chr(ord('A')+i)] = 0
```

```python
# 由于用例只有一个，不知道是否考虑完全
# 既然相似，那么单词长度一定相同，使用字母及其对应个数也应该相同
# 所以我们可以通过这样判断是否相似

# 判断单词长度是否相同
def check_len(a, b):
    if len(a) != len(b):
        return False
    return True

# 解析单词并返回字母及其对应个数，存储到字典中
def analysis_word(word):
    dic = dic_init.copy()
    for i in word:
        dic[i] += 1
    return dic

# 判断单词中字母及其对应个数是否相同
# ab 为对应字母字典
def check_word(a, b):
    for i in range(26):
        s = str(chr(ord('a')+i))
        l = str(chr(ord('A')+i))
        if a[s] != b[s] | a[l] != b[l]:
            return False
    return True

# 判断是否相似
def check_similar(a, b):
    if check_len(a, b):
        print(a + " len ok")
        if check_word(analysis_word(a), analysis_word(b)):
            print(a + " word ok")
            return True
    return False

# 找出相似单词
similar_words = []
for i in words:
    if check_similar(i, target):
        similar_words.append(i)

# 排序
similar_words.sort()
```

```python
for i in similar_words:
    print(i, end=' ')
```

```
bca word ok
abc bca (base) [root@cpy /]# /home/app/anaconda3/bin/python /home/prj/tes4.py
4
abc
dasd
tad
bca
abc
abc bca (base) [root@cpy /]#
```