题目描述：

开头和结尾都是元音字母（aeiouAEIOU）的字符串为 元音字符串 ，其中混杂的非元音字母数量为其 瑕疵度 。比如:

- “a” 、 “aa”是元音字符串，其瑕疵度都为0
- “aiur”不是元音字符串（结尾不是元音字符）
- “abira”是元音字符串，其瑕疵度为2

给定一个字符串，请找出指定瑕疵度的最长元音字符子串，并输出其长度，如果找不到满足条件的元音字符子串，输出0。

子串：字符串中任意个连续的字符组成的子序列称为该字符串的子串。

输入描述：

首行输入是一个整数，表示预期的瑕疵度flaw，取值范围[0, 65535]。

接下来一行是一个仅由字符a-z和A-Z组成的字符串，字符串长度(0, 65535]。

输出描述：

输出为一个整数，代表满足条件的元音字符子串的长度。

示例1

输入：

```
0

asdbuiodevauufgh
```

输出：

```
3
```

说明：

满足条件的最长元音字符子串有两个，分别为 *uio* 和 *auu*，长度为 3。

示例 2

输入：

```
2

aeueo
```

输出：

```
0
```

说明：

没有满足条件的元音字符子串，输出 0

示例 3

输入：

```
1
aabeebuu
```

输出：

```
5
```

说明：

满足条件的最长元音字符子串有两个，分别为 *aabee* 和 *eebuu*，长度为 5

```
def isy(ch):
    return ch in "aeiou" or ch in "AEIOU"
k = int(input())
s = input()
f = [isy(i) for i in s]
L, R, ans, cnt = 0, 0, 0, 0
while R < len(s):
    if not f[R]:
        cnt += 1
    else:
        while cnt > k or not f[L]:
            if not f[L]:
```

```
                cnt -= 1
            L += 1
        if cnt == k:
            ans = max(ans, R-L+1)
    R += 1
print(ans)
```