有时候人们会用额外的字母来表示额外的情感，比如 "hello" -> "heeellooo", "hi" -> "hiii"。我们将连续的相同的字母分组，并且相邻组的字母都不相同。我们将一个拥有三个或以上字母的组定义为扩张状态（extended），如第一个例子中的 "e" 和 "o" 以及第二个例子中的 "i"。 此外，"abbcccaaaa" 将有分组 "a"，"bb"，"ccc"，"dddd"；其中 "ccc" 和 "aaaa" 处于扩张状态。

对于一个给定的字符串 S ，如果另一个单词能够通过将一些字母组扩张从而使其和 S 相同，我们将这个单词定义为可扩张的（stretchy）。我们允许选择一个字母组（如包含字母 `c` ），然后往其中添加相同的字母 `c` 使其长度达到 3 或以上。注意，我们不能将一个只包含一个字母的字母组，如 "h"，扩张到一个包含两个字母的组，如 "hh"；所有的扩张必须使该字母组变成扩张状态（至少包含三个字母）。

输入一组单词，输出其中可扩张的单词数量。

示例:

输入:

```
S = "heeellooo"
words = ["hello", "hi", "helo"]
```

输出: 1

解释:

我们能通过扩张"hello"的"e"和"o"来得到"heeellooo"。

我们不能通过扩张"helo"来得到"heeellooo"因为"ll"不处于扩张状态。

说明：

- `0 <= len(S) <= 100`。
- `0 <= len(words) <= 100`。
- `0 <= len(words[i]) <= 100`。
- `S` 和所有在 `words` 中的单词都只由小写字母组成。

思路，该题的判断规则为，字符串的连续字母，若是字母大于等于3，则和单词比字母数目多少，若是单词多，则不能扩充出来，小于，则可以。若是字符串的单词个数小于3，则和单词比字母个数是否相等，相等则符合。然后，还需要判定的为单词字母是否是相等的。

```
class Solution {
public:
    int expressiveWords(string S, vector<string>& words) {
        if(S.size()==0){
            return 0;
        }
        string s="";
```

```
int num=1;
vector<int> number;
int result=0;
for(int i=0;i<S.size();i++){
    if(S[i]==S[i+1]){
        num++;
    }else{
        number.push_back(num);
        s+=S[i];
        num=1;
    }
}
for(int i=0;i<words.size();i++){
    int wz=0;
    num=1;
    for(int j=0;j<words[i].size();j++){
        if(s[wz]==words[i][j]){
            if(words[i][j]==words[i][j+1]){
                num++;
            }else{
                if(number[wz]>=3){
                    if(number[wz]>=num){
                        num=1;
                        wz++;
                    }else{
                        break;
                    }
                }else{
                    if(number[wz]==num){
                        num=1;
                        wz++;
                    }else{
                        break;
                    }
                }
```

```
                    }
                }else{
                    break;
                }
                if(wz==s.size()){
                    if(j==words[i].size()-1){
                        result++;
                    }else{
                        break;
                    }
                }
            }
        }
        return result;
    }
};
```