Emanuel estava estudando Programação Competitiva, resolvendo problemas sobre palíndromos. Um palíndromo é uma string que permanece igual quando lida de trás para frente, como `arara` e `tenet`. Além disso, Emanuel sabe que uma substring de uma string T é uma string que pode ser obtida apagando-se caracteres do início e fim de T. Por exemplo, `ara`, `ar` e `arara` são substrings de `arara`.

Ao se deparar com o problema clássico de computar quantas substring de uma string S são palíndromos, Emanuel se perguntou como que esse problema poderia ser resolvido se pudessemos re-ordenar cada substring. Ou seja, dado uma string S, quantas substrings de S (contando repetições) podem ser re-ordenadas para formar um palíndromo.

Cansado após implementar o problema clássico, Emanuel pede sua ajuda para implementar a variação para ele. Para simplificar o problema para você, ele garante que a string nunca conterá vogais (`a`, `e`, `i`, `o`, `u`, `y`).

Input

A primeira linha contém um inteiro N onde $1 \leq N \leq 10^6$. A segunda linha contém uma string S com N caracteres. S é composta por letras minúscula e não possui os caracteres `a`, `e`, `i`, `o`, `u`, `y`.

Output

Imprima um único inteiro: o número de substrings de S que podem ser re-ordenadas para formar um palíndromo.

Sample input 1	**Sample output 1**
5 ffggh	12

Explicação do Exemplo 1: As 12 substrings são:

1. `f`
2. `f`
3. `g`
4. `g`
5. `h`
6. `ff`
7. `gg`
8. `ffg`
9. `fgg`
10. `ggh`
11. `ffgg`
12. `ffggh`