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

The input consists of a single line that contains A primeira linha contém um inteiro N.

Input

The input consists of a single line that contains A segunda linha contém uma string S com N caracteres. S 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 eennt	12

Explicação do Exemplo 1: As 12 substring são:

1. `e`
2. `e`
3. `n`
4. `n`
5. `t`
6. `ee`
7. `nn`
8. `een`
9. `enn`
10. `nnt`
11. `eenn`
12. `eennt`