TAA - LEA 01

A. Divisibilidade por 11

4 seconds, 256 megabytes

Na matemática, existem diferentes métodos para verificar se um número é divisível por 11 ou não. Um deles é o seguinte: Coloca-se sinais alternados entre os algarismos, começando com o sinal positivo e, então, soma-se todos esses valores. Se o resultado da soma dessa série for múltiplo de 11 (incluindo o zero) então o número é divisível por 11.

Exemplos:

$617694 \rightarrow \ +6-1+7-6+9-4=11 \qquad \checkmark$

$111 \rightarrow \ +1-1+1=1 \qquad \times$

$9191919 \rightarrow \ +9-1+9-1+9-1+9=33 \qquad \checkmark$

Reorganizando a soma dos dígitos de forma a visualizar primeiro a posições pares (iniciando do zero) e em seguida o grupo dos dígitos nas posições ímpares, pode-se realizar a mesma verificação da seguinte forma:

$$\underline{9}2\ \underline{7}3\ \underline{5}4\ \underline{3}5\ \underline{1}6\ \underline{2}7\ \underline{4}8\ \underline{2}9 \ \rightarrow (9+7+5+3+1+2+4+2)-(2+3+4+5+6+7+8+9)=33-44=-11 \qquad \checkmark$$

Assim, sua tarefa é, dado um número, dizer se este é divisível por onze ou não.

Input

A entrada contém diversos casos de testes. Cada caso de teste é composto por duas linhas, onde a primeira linha contém um inteiro D ($1 \le D \le 100000$) que indica o número de dígitos do número e a segunda linha contém um inteiro N ($0 \le N \le 10^D$) que é o número a ser verificado. A entrada será finalizada se o valor D lido, for igual a -1.

Output

A saída deverá conter uma linha para cada valor verificado. Cada linha dever conter a frase "N: A - B = S - T", onde A e B são os somatórios dos dígitos em posições pares e ímpares, respectivamente, S é a soma de A e B e T é uma string contendo a palavra "sim" ou "não" indicando se o número é divisível por 11 ou não, conforme os exemplos.

input
```
6
617694
-1
```
output
```
617694: 22 - 11 = 11 - sim
```

input
```
3
111
16
9273543516274829
7
9191919
7
1919191
-1
```
output
```
111: 2 - 1 = 1 - nao
9273543516274829: 33 - 44 = -11 - sim
9191919: 36 - 3 = 33 - sim
1919191: 4 - 27 = -23 - nao
```

A quantidade de testes, em um único caso de testes, não será superior a 100.

B. Fibonacci?

1 second, 256 megabytes

Todo mundo conhece a sequência de Fibonacci, mas quase ninguém ouviu da sua amiga mais próxima, que é a sequência de lambda. Ela é uma sequência que compreende os denominadores das aproximações racionais mais próximas da raiz quadrada de 2. Por sorte (para a nossa sorte), ela pode ser expressa por uma função, bem parecida com a sequência de fibonacci, dada pelo seguinte:

$$P(n) = \begin{cases} 0 & \text{se } n = 0; \\ 1 & \text{se } n = 1; \\ 2P(n-1) + P(n-2) & \text{caso contrário.} \end{cases}$$

Assim, sua tarefa é escrever um programa que, dado um determinado inteiro, calcule a função de lambda desse número.

Input

A primeira linha da entrada contém um número C ($1 \le C \le 10$) que indica a quantidade de casos de testes. As C linhas seguintes contém um inteiro N cada uma ($0 \le N \le 32$) que é o valor cujo a função lambda deve ser calculada.

Output

A saída deve conter C linhas, onde cada linha contém um inteiro que é o respectivo resultado da função lambda, conforme os exemplos.

input
```
1
3
```
output
```
5
```

input
```
2
5
8
```
output
```
29
408
```

No primeiro caso de testes ($P(3)$), os passos são os seguintes:

```
P(3)  = 2 * P(2) + P(1)
      = 2 * (2 * P(1) + P(0)) + 1
      = 2 * (2 * 1 + 0) + 1
      = 2 * 2 + 1
      = 5
```

Observação importante: Linguagens interpretadas (aka python ou java) provavelmente não irão passar dentro do tempo.

Observação importante 2: Lembre-se de testar a sua solução com os casos de teste limite para o problema.

C. Primos Gêmeos

1 second, 256 megabytes

Um número inteiro primo é aquele que tem '"somente"' quatro divisores distintos, $p \in Z$: ± 1 e $\pm p$. Já um número natural primo tem '"unicamente"' dois divisores naturais distintos: o número um e ele mesmo.

Além disso, na matemática, existem diversas outras propriedades advindas dos números primos. Uma delas, por exemplo, são os **números primos gêmeos**. Dois números são considerados primos gêmeos se ambos são primos, e a diferença entre eles é de apenas 2 posições. São exemplos de primos gêmeos:

$(3, 5),\ (5, 7),\ (11, 13),\ (17, 19),\ (29, 31)$

Há inclusive uma conjectura que diz que há infinitos primos gêmeos, mas essa será uma discussão para outra hora. Nesta tarefa, estamos interessados em saber apenas se, dado um número inteiro, se ele é um primo gêmeo ou não.

Input

A primeira linha da entrada contém um inteiro N ($1 \leq N \leq 10000$), que indica a quantidade de números a serem verificados. Em seguida haverá N linhas onde cada linha contem um inteiro V ($1 \leq V \leq 10^6$) com cada um dos valores a serem verificados.

Output

Para cada inteiro V, imprima uma linha de saída com a mensagem "O numero X eh um primo gemeo" ou "O numero X nao eh um primo gemeo", onde X é o inteiro verificado de acordo com a entrada e a especificação do problema.

input

```
3
5
7
9
```

output

```
O numero 5 eh um primo gemeo
O numero 7 eh um primo gemeo
O numero 9 nao eh um primo gemeo
```

input

```
6
11
23
31
33
2
3
```

output

```
O numero 11 eh um primo gemeo
O numero 23 nao eh um primo gemeo
O numero 31 eh um primo gemeo
O numero 33 nao eh um primo gemeo
O numero 2 nao eh um primo gemeo
O numero 3 eh um primo gemeo
```

Repare que 2 e 3 são primos e se diferem em apenas uma posição, então os mesmos não considerados primos gêmeos, porém o 3 é gêmeo do 5.