TAA - LEE 01

A. Número Faltante

1 second, 256 megabytes

Uma permutação de N inteiros é uma reorganização específica dos números inteiros de 1 a N, de modo que cada número apareça exatamente uma vez na sequência resultante. Por exemplo, se utilizarmos $N = 10$, uma permutação válida de N seria:

$$9, 4, 2, 7, 10, 1, 6, 8, 3, 5$$

Neste exercício, dada uma permutação de N inteiros, **um inteiro dessa permutação foi removido** e sua tarefa é identificar qual é esse número faltante dessa permutação.

Input

A primeira linha contém um inteiro N ($2 \le N \le 100000$), que representa o tamanho da permutação.

A segunda linha possui $N - 1$ inteiros, separados por espaço, que são a lista de números a qual o número faltante deverá ser identificado.

Output

A saída deverá conter um único inteiro que é o número faltante dessa permutação.

input
```
10
3 8 2 4 1 10 6 7 9
```
output
```
5
```

input
```
12
3 5 10 2 4 11 1 12 6 7 9
```
output
```
8
```

B. Divisibilidade por 3

3 seconds, 256 megabytes

Na matemática, um número é divisível por 3 quando a soma dos valores absolutos de seus algarismos resultar em um número divisível por 3. Nesse caso, o resto será o mesmo que o deixado na divisão da soma dos valores absolutos do número por 3.

Exemplos:

$51 \to 5 + 1 = 6 \quad \checkmark$

$101 \to 1 + 0 + 1 = 2 \quad \times$

$234 \to 2 + 3 + 4 = 9 \quad \checkmark$

$7.851 \to 7 + 8 + 5 + 1 = 21 \to 2 + 1 = 3 \quad \checkmark$

$9.631 \to 9 + 6 + 3 + 1 = 19 \to 1 + 9 = 10 \to 1 + 0 = 1 \quad \times$

$998.877.665.544 \to 9 + 9 + 8 + 8 + 7 + 7 + 6 + 6 + 5 + 5 + 4 + 4 = 78 \to 7 + 8 = 15 \to 1 + 5 = 6 \quad \checkmark$

Assim, sua tarefa é, dado um número, dizer se este é divisível por três 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 um inteiro que é a soma dos dígitos e a palavra `sim` se o número for divisível por 3, ou `nao` caso contrário, conforme os exemplos.

input
```
2
15
-1
```
output
```
6 sim
```

input
```
3
300
3
100
-1
```
output
```
3 sim
1 nao
```

A quantidade de testes em um único caso de testes, não será superior a 100. Além disso, tenha certeza de ter entendido os limites (entrada) da questão. Por último, a soma do dígitos de um número deve ser feita somente uma vez (por exemplo, o número 888 resulta em 24 e não é necessário realizar o procedimento novamente de somar os dígitos do número 24).

C. BODH

1 second, 256 megabytes

Jean está aprendendo sobre bases numéricas e para isso ele resolveu realizar um monte de conversões de bases para praticar. Seu professor ensinou alguns macetes para facilitar a conversão de bases, mas Jean ainda não está muito familiarizado com eles.

Quando ele recebe um número em base decimal, ele lembra que precisa escrever, da direita para a esquerda, todas as potências de 2 a partir de 0, até a quantidade de bits que o tipo numérico que ele está utilizando suporta. Por exemplo, se ele estiver utilizando um tipo numérico de 8 bits e quiser converter o número 115, ele deve escrever as potências de 2 da seguinte forma:

```
___  ___  ___  ___  ___  ___  ___  ___
128   64   32   16    8    4    2    1
```

e a partir disso, ele deve marcar as potências de 2 que somadas resultam no número que ele deseja converter. No caso do número 115, ele deve marcar as potências de 2 que somadas resultam em 115, que são 64, 32, 16, 2 e 1. O resultado final fica:

```
 0    1    1    1    0    0    1    1
___  ___  ___  ___  ___  ___  ___  ___
128   64   32   16    8    4    2    1
```

Então, o número 115 em base decimal é igual a 01110011 em base binária. Já para a conversão para as bases octal e hexadecimal, ele aprendeu que para facilitar deve sempre converter primeiro o número para base binária e depois para a base desejada. No caso da conversão para a base hexadecimal, ele deve agrupar os bits de 4 em 4, da direita para a esquerda, e converter cada grupo para um número hexadecimal. No caso do número 115, já convertido para binário, os bits agrupados ficam:

```
|  0    1    1    1  |  0    0    1    1  |
|___  ___  ___  ___  | ___  ___  ___  ___
|  8    4    2    1  |  8    4    2    1  |

                  7                    3
```

Dessa forma, somando-se os valores de cada grupo, temos que 0111 é igual a 7 e 0011 é igual a 3, então o número 115 em base hexadecimal é igual a 73. Já para converter para a base octal, o raciocínio é bem parecido, mas agrupando-se os bits de 3 em 3.

```
|  0    0    1  |  1    1    0  |  0    1    1  |
|___  ___  ___  | ___  ___  ___ | ___  ___  ___
|  4    2    1  |  4    2    1  |  4    2    1  |
   |
   |         1              6              3
   |
```

```
    —- posição 9 apenas para fins de organização e
visualização
```

Assim, o número 115 em base octal é igual a 163.

Como Jean precisa dos gabaritos para conferir as questões que ele praticou, ele pediu sua ajuda para escrever um programa que dado um número em base decimal, o ajude a converter para as bases binária, octal e hexadecimal.

Input

A entrada contém múltiplos casos de testes. Cada linha possui dois inteiros N $(0 \leq N < 2^{63})$ e B $(2 \leq B \leq 16)$, onde N é o número em base decimal que Jean deseja converter e B é a base para a qual ele deseja converter o número, podendo ser 2, 8, 10 ou 16. A entrada termina com os valores $N = B = -1$.

Output

Para cada caso de teste, imprima uma linha com a conversão do número N para a base B, conforme os exemplos.

input

```
115 2
115 8
115 10
115 16
-1 -1
```

output

```
1110011
163
115
73
```

input

```
42 2
85 2
85 16
170 16
170 2
1337 10
-1 -1
```

output

```
101010
1010101
55
AA
10101010
1337
```

Lembre-se que a linguagem C não tem uma conversão de decimal para binário, por padrão. Para isso, escreva uma função para tal. Caso nunca o tenha feito antes o link a seguir pode te auxiliar: https://www.todamateria.com.br/numeros-binarios/.

Aproveite a oportunidade para exercitar o seu entendimento e lógica sobre o conteúdo.

D. É primo ou não é

1 second, 256 megabytes

'''Número primo''' é qualquer número p cujo conjunto dos divisores não inversíveis não é vazio, e todos os seus elementos são produtos de p por números inteiros inversíveis. De acordo com esta definição, 0, 1 e -1 não são números primos.

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

Nesta tarefa, estamos interessados em saber, dada uma lista de inteiros naturais, quais destes números são primos e quais não são.

Input

A primeira linha da entrada contém um inteiro N $(1 \leq N \leq 1000)$, que indica a quantidade de números a serem verificados. Em seguida haverá N linhas onde cada linha contem um inteiro V $(1 \leq V \leq 10^7)$ com cada um dos valores cujo a primalidade deverá ser verificada.

Output

Para cada inteiro V, imprima uma linha de saída com a mensagem "o numero X eh primo" ou "o numero X nao eh primo", 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 primo
o numero 7 eh primo
o numero 9 nao eh primo
```

input

```
6
5
5
15
51
19
32
```

output

```
o numero 5 eh primo
o numero 5 eh primo
o numero 15 nao eh primo
o numero 51 nao eh primo
o numero 19 eh primo
o numero 32 nao eh primo
```

E. Função de McCarthy

1 second, 256 megabytes

John McCarthy (não confundir com John McAfee) foi um cientista da computação muito famoso, por ser um dos fundadores da disciplina de Inteligência Artificial.

Ele trabalhou por muito tempo com teoria da computação e, em um de seus trabalhos, ele definiu uma função recursiva bem simples, para ser utilizada em testes de verificação formal. A função definida por ele, é a seguinte:

$$M(n) = \begin{cases} n - 10 & \text{se } n > 100 \\ M(M(n+11)) & \text{se } n \leq 100 \end{cases}$$

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

Input

A primeira linha da entrada contém um número C $(1 \leq C \leq 100000)$ que indica a quantidade de casos de testes. As C linhas seguintes contém um inteiro N cada uma $(0 \leq N \leq 10000000)$ que é o valor cujo a função McCarthy deve ser calculada.

Output

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

input

```
1
98
```

output

```
91
```

input

```
2
35
141
```

output

```
91
131
```

Na única entrada do primeiro caso de testes ($M(98)$), os passos são os seguintes:

```
M(98) = M(M(109))
      = M(99)
      = M(M(110))
      = M(100)
      = M(M(111))
      = M(101)
      = 91
```

F. Problema Fácil que Ninguém Resolveu

1 second, 256 megabytes

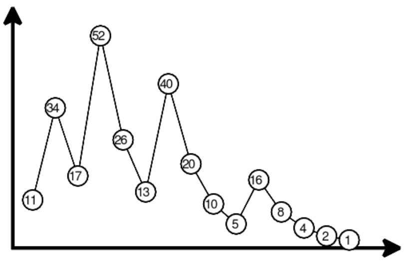
Em 1937, o alemão Lothar Collatz, formulou uma hipótese matemática que segue, até hoje, não demonstrada. Pela sua simplicidade, a conjectura tem atraído matemáticos profissionais e amadores, que tentam provar a sua veracidade. Muitos cientistas passaram anos, décadas, tentando resolvê-lo, inutilmente. Durante a guerra a fria, se dizia até que o problema foi inventado pelos soviéticos para atrasar a ciência nos EUA.

Na Ciência da Computação, problemas costumam ser classificados como pertencentes a uma determinada classe de problemas (por exemplo, NP, em aberto, recursivo, etc). Neste exercício você estará analisando uma propriedade de um algoritmo cuja classificação não é conhecida para todas as entradas possíveis, que é o proposto por Collatz e ficou conhecido como Conjectura de Collatz.

A ideia por trás da conjectura é bem simples, dado um inteiro N, há duas regras a serem aplicadas:

1. Se N for par, divida por 2 $\rightarrow \frac{N}{2}$
2. Se N for ímpar, multiplique por 3 e adicione 1 $\rightarrow 3N+1$

O objetivo é aplicar sucessivamente essas regras até que o resultado seja 1. Por exemplo, para $N = 11$, a aplicação sucessiva das regras resulta no seguinte:



A conjectura diz que para qualquer número natural inteiro, a aplicação sucessiva das regras sempre terminará em 1.

Assim, sua tarefa é, dado um inteiro, aplicar as regras apresentadas sucessivamente até que o valor seja 1.

Input

A entrada possui um inteiro N ($1 \leq N \leq 10^6$), que é o valor inicial.

Output

A saída deve conter a sequência de inteiros, separados por espaço, da aplicação das regras da conjectura até o valor 1, conforme os exemplos.

input
5
output
5 16 8 4 2 1

input
11
output
11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Dica: 113383

Curiosidade (não é necessário para resolver o exercício): https://www.youtube.com/watch?v=094y1Z2wpJg