

Universidade Federal de Goiás – UFG  
Instituto de Informática – INF  
**Bacharelados (Núcleo Básico Comum)**  
**Algoritmos e Estruturas de Dados 1**  
**Lista de Exercícios 01 – Revisão**  
Elaborada pelo Prof. Wanderley de Souza Alencar

## Sumário

<b>1</b>	<b>cometa (+)</b>	<b>2</b>
<b>2</b>	<b>José (+)</b>	<b>4</b>
<b>3</b>	<b>overflow (++)</b>	<b>6</b>
<b>4</b>	<b>capicua (++)</b>	<b>8</b>
<b>5</b>	<b>Computação (++)</b>	<b>10</b>
<b>6</b>	<b>Envelopes (++)</b>	<b>12</b>
<b>7</b>	<b>sapos (+++)</b>	<b>15</b>
<b>8</b>	<b>primos (++)</b>	<b>18</b>
<b>9</b>	<b>vetores (+++)</b>	<b>20</b>
<b>10</b>	<b>cálculo de áreas (+++)</b>	<b>22</b>
<b>11</b>	<b>manipulação de matrizes 1 (++)</b>	<b>24</b>
<b>12</b>	<b>manipulação de matrizes 2 (++++)</b>	<b>27</b>
<b>13</b>	<b>Números de Fibonacci (++)</b>	<b>29</b>
<b>14</b>	<b>Fatoração de Números de Fibonacci (++++)</b>	<b>31</b>



## 1 cometa (+)



O cometa Halley é um dos cometas de menor período do Sistema Solar, completando uma volta em torno do Sol a cada 76 anos. Na última ocasião em que ele se tornou visível do planeta Terra, em 1986, várias agências espaciais enviaram sondas para coletar amostras de sua cauda e assim confirmar teorias sobre sua composição química. Saiba mais sobre ele em <http://astro.if.ufrgs.br/solar/halley.htm>.

Escreva um programa C que, dado o ano atual, determina qual o próximo ano em que o cometa Halley será visível novamente no planeta Terra. Se o ano atual é um ano de passagem do cometa, considere que o cometa já passou nesse ano, ou seja, considere sempre o próximo ano de passagem após o atual.

**Observação:** Não se esqueça de considerar os anos bissextos, ou seja, que a cada quatro anos (em direção ao futuro ou ao passado) há um *erro* de um dia em relação ao ano solar que, neste caso, é considerado como tendo exatamente 365 dias terrestres. O ano de 1986, quando o cometa de Halley se tornou visível na Terra pela última vez, é considerado o “*marco de sincronismo*” para os cálculos do programa a ser elaborado.

### Entrada

A única linha da entrada do programa contém um único inteiro  $A$ , indicando o ano atual, sendo que  $0 \leq A \leq 10^4$ .

### Saída

Seu programa deve imprimir uma única linha, contendo um número inteiro, indicando o próximo ano em que o cometa Halley será visível novamente do planeta Terra.

### Exemplos

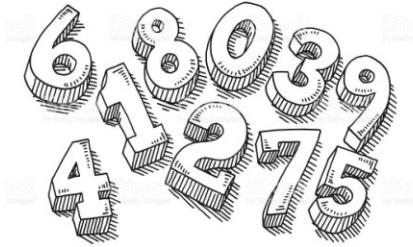
Entrada	Saída
2635	2670

<b>Entrada</b>	<b>Saída</b>
2010	2062

<b>Entrada</b>	<b>Saída</b>
2270	2290

<b>Entrada</b>	<b>Saída</b>
1910	1986

<b>Entrada</b>	<b>Saída</b>
460	465



## 2 José (+)



(+)

João tem um irmão mais novo, José, que começou a ir à escola e já está tendo problemas com números. Para ajudá-lo a “*pegar o jeito*” com a escala numérica, a professora de José escreve dois números de três dígitos e pede ele para comparar esses números.

Entretanto, ao invés de interpretá-los com o dígito mais significativo à esquerda, ele deve interpretá-los com o dígito mais significativo à direita. Ele tem que dizer à professora qual o maior dos dois números.

Escreva um programa em C que seja capaz de verificar as respostas de José.

### Entrada

A entrada conterá uma única linha com dois números de três dígitos, A e B, os quais não serão iguais.

### Saída

A saída deve conter, uma linha com o maior dos números na entrada, comparados conforme descrito no enunciado da tarefa. O número deve ser escrito invertido, para mostrar a José como ele deve lê-lo e, havendo zeros à esquerda do número, eles não devem ser escritos, como mostra o quatro exemplo a seguir.

### Exemplos

Entrada	Saída
483 583	385

Entrada	Saída
493 583	394

Entrada	Saída
493 589	985

<b>Entrada</b>	<b>Saída</b>
160 720	61

<b>Entrada</b>	<b>Saída</b>
100 200	2



### 3 overflow (++)



Os computadores digitais foram inventados para realizar cálculos muito rapidamente e, atualmente, atendem a esse requisito de maneira extraordinária. Porém, nem toda “*conta*” pode ser feita num computador digital típico, pois ele não consegue representar todos os números dentro de um único endereço de memória.

Num típico, e simples, computador pessoal atual, por exemplo, o maior inteiro que é possível representar numa unidade de sua memória é  $18.446.744.070.000.000.000$  ( $2^{64} - 1$ ). Caso alguma “*conta*” executada pelo computador dê um resultado acima desse número, ocorrerá o que chamamos de *overflow*, que é quando o computador faz uma “*conta*” e o resultado não pode ser representado por ser maior do que o valor máximo permitido (em inglês *overflow* significa *trasbordar*).

Por exemplo, se um computador fictício somente pode representar números menores ou iguais a  $1023$  ( $2^{10} - 1$ ) e mandarmos ele executar a conta  $1022 + 5$ , vai ocorrer um *overflow*, já que o resultado deste cálculo é maior que  $1023$ .

Elabore um programa C que seja capaz de receber o maior número que um computador consegue representar em sua memória e uma expressão de soma ou de multiplicação entre dois inteiros positivos, determine se ocorrerá, ou não, *overflow* naquele computador.

#### Entrada

A primeira linha da entrada contém um inteiro  $N$  representando o maior número que o computador consegue representar.

A segunda linha contém um inteiro  $N_1$ , seguido de um espaço em branco, de um caractere  $C$  (que pode ser ‘+’ ou ‘x’, representando os operadores de *adição* e de *multiplicação*, respectivamente), de outro espaço em branco, e, finalmente, de outro número inteiro  $N_2$ .

Assim, a segunda linha da entrada representa a expressão  $N_1 + N_2$ , se o caractere C for ‘+’, ou  $N_1 \times N_2$ , se o caractere C for ‘x’.

#### Saída

Seu programa deve imprimir a palavra ‘*overflow*’ se o resultado da expressão causar um *overflow* no computador, ou a expressão ‘*no overflow*’ caso contrário.

Ambas as palavras devem ser escritas com todas as letras minúsculas.

#### Exemplos

<b>Entrada</b>	<b>Saída</b>
57 20 x 3	overflow

<b>Entrada</b>	<b>Saída</b>
10 5 + 6	overflow

<b>Entrada</b>	<b>Saída</b>
10 5 + 4	no overflow

<b>Entrada</b>	<b>Saída</b>
57 12 x 3	no overflow

<b>Entrada</b>	<b>Saída</b>
30 4 x 4	no overflow



## 4 capicua (++)



(++)

O pequeno estudante Alan Mathison Turing está aprendendo a decompor um número em unidades, dezenas, centenas, unidade de milhar, dezena de milhar, etc. Ele está com grandes dificuldade neste processo. Sua professora, Ada Lovelace, preocupada com o rendimento de Alan decidiu ensiná-lo por meio de uma brincadeira:

Alan deve pegar um número com quatro algarismos e verificar se o reverso deste número é ele próprio. Se for, Alan deve responder *yes*, do contrário deve responder *no* – Alan é britânico e, por isso, responde em inglês.

Em verdade, Ada está ensinando quando um número é chamado de *capicua* (ou também conhecido por *palíndromo*), pois um número é dito *capicua* quando seu reverso é ele próprio.

Escreva um programa C que implemente esta brincadeira conforme descrito a seguir.

### Entrada

A primeira linha da entrada contém um inteiro  $N$  ( $N \geq 1$ ) representando a quantidade de números inteiros que Alan deve responder *yes* (*capicua*) ou *no*. Cada uma das  $N$  linhas seguintes será composta por um inteiro de até quatro algarismos.

**Observação:** O seu programa não poderá decompor o número na entrada, ou seja, não poderá ler o número  $N$  como caracteres individuais que formam o número: você deve fazer o programa lê-lo como número e, em seguida, manipulá-lo da maneira que desejar para que possa gerar a resposta correta.

### Saída

A saída consiste de uma única linha: a sequência de palavras *yes/no*, separadas por um único espaço em branco entre cada par consecutivo delas, que corresponde à resposta.

### Exemplos

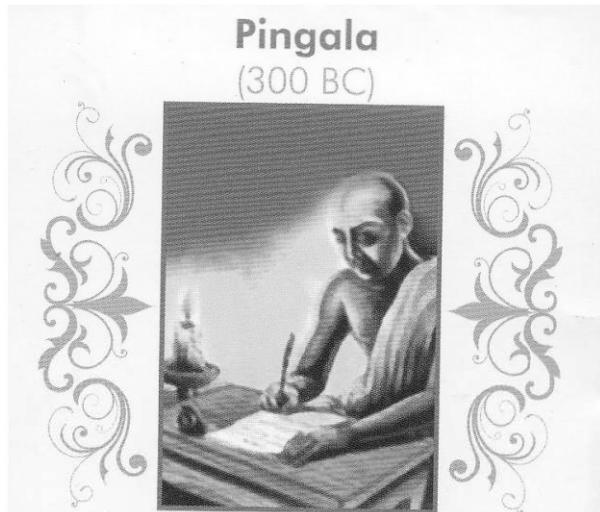
Entrada	Saída
2 4569 5775	no yes

<b>Entrada</b>	<b>Saída</b>
3 1458 1228 9779	no no yes

<b>Entrada</b>	<b>Saída</b>
4 1221 2222 3113 7887	yes yes yes yes

<b>Entrada</b>	<b>Saída</b>
5 1234 1243 1324 2134 4321	no no no no no

<b>Entrada</b>	<b>Saída</b>
10 1234 5897 1881 2662 5588 9229 3653 5555 3773 4587	no no yes yes no yes no yes yes no



## 5 Computação (++)



(++)

A capacidade natural do ser humano para calcular quantidades, nos mais variados modos, foi um dos fatores que possibilitaram o desenvolvimento da Matemática, da Lógica e, por conseguinte, da Computação. Nos primórdios da Matemática e da Álgebra, utilizavam-se os dedos das mãos para efetuar cálculos, daí a origem da palavra *dígito*.

Por volta do século III a.C., o matemático indiano Pingala inventou o sistema de numeração binário, que ainda hoje é utilizado no processamento de todos computadores digitais: o sistema estabelece que sequências específicas de 1's (uns) e 0's (zeros) pode representar qualquer número, letra, imagem, etc.

Entretanto, a Computação está evoluindo rapidamente e recentemente a SBC (Sociedade Brasileira de Computação) inventou um computador com a base 4 (tetrade), inspirado na Biologia (lembra-se? Adenina, Citosina, Guanina e Timina, as quatro bases nitrogenadas!).

A SBC contratou você para fazer um programa C que receba um número inteiro positivo, na base decimal, e converta-o para a base 4 utilizando divisões sucessivas. Você deve escrever um programa que, a partir de uma lista de números, calcule o valor correspondente de cada desses números na base 4.

**Observação:** Considere que os símbolos utilizados para representar as quantidades ZERO, UM, DOIS e TRÊS, na base 4 são, respectivamente, A, C, G e T.

### Entrada

A entrada contém um único conjunto de testes, que deve ser lido do dispositivo de entrada padrão (o teclado).

A primeira linha contém o número de inteiros  $N$  ( $1 \leq N \leq 100$ ) que será digitada.

A segunda linha contém  $N$  números inteiros  $n_i$ , cada um representando um número decimal.

### Saída

Seu programa deve imprimir, na saída padrão, os valores correspondentes na base 4 – um por linha – para cada número decimal digitado.

### Exemplos

**Observação:** Note que as letras (A, C, G, T) são sempre grafadas em maiúsculas.

<b>Entrada</b>	<b>Saída</b>
5 1 2 3 4 10	C G T CA GG

<b>Entrada</b>	<b>Saída</b>
2 16 8	CAA GA

<b>Entrada</b>	<b>Saída</b>
9 1 2 3 4 10 16 8 5 11	C G T CA GG CAA GA CC GT

<b>Entrada</b>	<b>Saída</b>
5 10 20 30 40 50	GG CCA CTG GGA TAG

<b>Entrada</b>	<b>Saída</b>
10 1 2 3 4 1 2 3 4 1 2	C G T CA C G T CA C G



## 6 Envelopes (++)



(++)

Kurt Gödel é um garoto muito esperto e que adora promoções e sorteios. Como já participou de muitas promoções da forma “para participar, envie  $n$  rótulos de produtos ...”, Kurt tem o hábito de guardar o rótulo de todos os produtos que compra prevendo a possibilidade de ocorrência de promoções futuras. Dessa forma, sempre que uma empresa faz uma promoção ele já tem “*um monte*” de rótulos para mandar. A SBC (Super Balas e Caramelos Ltda) está fazendo uma nova SUPER promoção, e, como era de se esperar, Kurt quer participar dela: é preciso enviar um envelope contendo um rótulo de cada tipo de bala que a SBC produz.

Por exemplo, se a SBC disser que produz três tipos de balas (A, B, C) e uma pessoa tem três rótulos de A, três rótulos de B e dois rótulos de C, ela pode enviar no máximo dois envelopes, já que falta um rótulo de C para compor o terceiro envelope.

Sabe-se que não há limite para o número de envelopes que uma pessoa pode enviar para a promoção da SBC. Balas são a segunda coisa de que Kurt mais gosta (a primeira, como você, sabe são as *promoções*) e, por causa disso, a quantidade de rótulos que ele tem é muito grande: ele não está conseguindo determinar a quantidade máxima de envelopes que pode enviar.

Como você é o(a) melhor amigo(a) de Kurt, ele pediu sua ajuda para fazer o cálculo, de modo que ele compre o número exato de envelopes necessários para enviar para esta promoção.

Você deve escrever um programa C que, a partir da lista de rótulos de Kurt, calcule o número máximo de envelopes válidos que ele pode enviar para a promoção da SBC.

### Entrada

A entrada contém um único conjunto de testes, que deve ser lido do dispositivo de entrada padrão (o teclado).

A primeira linha contém dois números inteiros  $N$  e  $K$  representando, respectivamente, a quantidade de rótulos de balas que Kurt possui e o número de tipos diferentes de bala que a SBC produz. Os tipos de balas são identificados por inteiros de 1 a  $K$ , com  $1 \leq N \leq 1000$  e  $1 \leq K \leq 20$ .

A segunda linha contém  $N$  números inteiros, digamos  $n_i \in \{1, 2, 3, \dots, K\}$ , cada um representando um rótulo de bala que Kurt possui.

### Saída

Seu programa deve imprimir, na saída padrão, o número máximo de envelopes válidos que Kurt pode enviar.

## Exemplos

Entrada	Saída
10 2 1 1 1 1 1 2 2 2 2 2	5

Entrada	Saída
20 5 1 2 3 4 1 2 3 4 1 2 3 4 5 1 2 3 4 5 4 4	2

Entrada	Saída
10 3 1 2 3 1 2 3 1 2 3 1	3





## 7 sapos (+++)



(+++)

Sebastião Bueno Coelho, apelidado de SBC pelos familiares e amigos, passou as férias de janeiro de 2019 no sítio de seus avós. Durante sua estadia, uma das atividades prediletas do SBC era nadar no rio que havia no “fundo” da casa dos avós. Uma das características do rio que mais impressionava SBC era um belo caminho, feito inteiramente com pedras brancas.

Há muito tempo, o avô de SBC notara que os habitantes do sítio atravessavam o rio com grande frequência e, por isso, construiu um caminho nele feito com pedras posicionadas em linha reta; ao fazê-lo, tomou muito cuidado para que o espaçamento entre as pedras fosse exatamente de um metro. Hoje em dia, a única utilidade do caminho é servir de diversão para os sapos que vivem no rio, que pulam de uma pedra a outra agitadamente.

Um certo dia, enquanto descansava e nadava nas águas, SBC assistiu atentamente às acrobacias dos anfíbios Anura e notou que cada sapo sempre pulava uma quantidade fixa de metros.

SBC sabe que você participa, todos os anos, da *Maratona de Programação* do INF/UFG capitaneada pelo Prof. Humberto Longo, do INF/UFG, e chegando na escola, resolveu desafiar-lhe com o seguinte problema:

“Dado o número de pedras no rio –  $P$  –, o número de sapos –  $S$ , a pedra inicial sobre a qual cada sapo está, sabendo-se que cada pedra é identificada por sua posição na sequência de pedras a partir da margem do rio que está no “fundo” da casa dos avós de SBC –  $1, 2, 3, \dots$  – e, por fim, a distância que cada sapo pula –  $d_1, d_2, d_3, \dots, d_p$  –, determinar as posições onde pode existir pelo menos um sapo depois que SBC chega no rio após assistir ao balé dos pulos dos sapos.”.

### Entrada

A primeira linha da entrada contém dois inteiros  $P$  ( $1 \leq P \leq 50$ ) e  $S$  ( $1 \leq S \leq 100$ ) representando, respectivamente, o número de pedras no rio e o número de sapos.

Cada uma das  $S$  linhas seguintes possui dois inteiros  $p_i$  e  $d_i$  representando, respectivamente, a posição inicial de um sapo  $i$  e a distância fixa de pulo dele.

### Saída

A saída contém  $P$  linhas. A  $j$ -ésima linha indica a possibilidade, ou não, de ter um sapo na  $j$ -ésima pedra.

Para as pedras que podem ter um sapo você deve imprimir 1, e para as pedras que, com certeza, não podem ter nenhum sapo você deve imprimir 0.

### Exemplos

<b>Entrada</b>	<b>Saída</b>
5 2	1
3 2	0
4 4	1
	1
	1

<b>Entrada</b>	<b>Saída</b>
8 3	0
3 3	1
2 2	1
6 2	1
	0
	1
	0
	1

<b>Entrada</b>	<b>Saída</b>
10 8	1
1 7	1
2 5	1
3 4	1
4 7	1
5 2	1
6 9	1
7 2	1
8 3	1
	0

<b>Entrada</b>	<b>Saída</b>
1 6 7	1
1 8	1
2 7	1
3 6	1
4 5	1
5 4	1
6 3	1
7 2	0
	1
	0
	1
	1
	1
	1
	1
	1

<b>Entrada</b>	<b>Saída</b>
10 10	1
1 1	1
2 1	1
3 1	1
4 1	1
5 1	1
6 1	1
7 1	1
8 1	1
9 1	1
10 1	1



## 8 primos (++)



(++)

No livro *A música dos números primos*, de Marcus du Saboy (2007, Editora Zahar, 471 páginas), o autor mostra que o mistério dos *números primos* passou a ser considerado o maior problema matemático de todos os tempos. Em meados do século XIX, o alemão Georg Friedrich Bernhard Riemann (1826 – 1866) formulou uma hipótese:

“É possível estabelecer uma harmonia entre esses números primos, à semelhança da harmonia musical.” A partir de então, as mentes mais ambiciosas da Matemática embarcaram nessa procura que parece não ter fim. Atualmente, estipulou-se o prêmio de um milhão de dólares para quem provar a hipótese. O livro relata esse verdadeiro *Santo Graal* da Matemática, com casos interessantes e retratos pitorescos dos personagens que, desde Euclides, se envolveram nesse estranho mistério.

Você deverá, assim, pesquisar e implementar, em C, um algoritmo que seja capaz de identificar se um dado número inteiro positivo é, ou não, um *número primo*. Número que não é primo é denominado de *composto*.

### Entrada

A primeira linha da entrada contém um inteiro  $N$  ( $1 \leq N \leq 100$ ) representando a quantidade de números inteiros positivos para os quais seu programa deve responder *primo* ou *composto*.

Cada uma das  $N$  linhas seguintes será composta por um inteiro positivo.

**Observação:** O seu programa deve estar preparado para receber números no intervalo de 2 a  $2^{64} - 1$ .

### Saída

A saída consiste de  $N$  linhas, cada uma com a palavra *primo*, se o número for primo, ou a palavra *composto* caso o número não seja primo (número composto). Note que as palavras devem ser grafadas, necessariamente, com letras minúsculas.

### Exemplos

<b>Entrada</b>	<b>Saída</b>
5	primo
2	primo
3	primo
11	composto
16	composto
60	

<b>Entrada</b>	<b>Saída</b>
9	composto
1200	primo
1697	composto
2712	primo
2549	primo
4723	primo
7853	primo
23557	composto
23558	primo
15485863	

<b>Entrada</b>	<b>Saída</b>
10	primo
23	primo
29	primo
31	primo
37	primo
73	primo
79	primo
83	primo
101	primo
103	primo
107	

<b>Entrada</b>	<b>Saída</b>
10	composto
24	composto
30	composto
32	composto
38	composto
74	composto
80	composto
84	composto
102	composto
104	composto
108	



## 9 vetores (+++)



(+++)

Uma operação comum em diversas áreas da computação científica é a multiplicação de números inteiros positivos com grande número de dígitos. Por exemplo, multiplicar um número de 24 dígitos por outro de 16 dígitos, o que pode gerar um número de até 40 dígitos.

Você está participando de uma equipe de desenvolvimento de uma aplicação científica que deve implementar, utilizando o conceito de *vetor* para representar cada um dos números envolvidos, a operação de multiplicação mencionada.

A aplicação deve ser desenvolvida utilizando a linguagem C, conforme a seguir especificado.

### Entrada

A primeira linha da entrada conterá o número de casos de teste,  $t$ , a serem aplicados. Sabe-se que  $1 \leq t \leq 50$ .

A seguir são apresentadas  $t$  linhas, cada uma contendo os dois números inteiros a serem multiplicados, digamos  $m$  e  $n$ , sabendo-se que eles terão no máximo 40 dígitos cada, mas que também poderão ser iguais a 0 (zero). A dupla de números está separada por um único espaço em branco.

### Saída

A saída consiste de  $t$  linhas, cada uma com o resultado da operação de multiplicação dos pares de números correspondentes, na ordem em que foram fornecidos.

### Exemplos

Entrada	Saída
1 9423891297239 123857601272	1167220570724098896488008

**Observação:** Devido ao comprimento dos números envolvidos, os exemplos podem ter mais linhas impressas que aquelas registradas nos dados de entrada fornecidos (veja o 2º exemplo). Sempre considere que cada *caso* é fornecido numa única linha, com os números  $m$  e  $n$  sendo fornecidos numa única linha e separados por um único espaço em branco entre eles.

<b>Entrada</b>	<b>Saída</b>
2 9423891297239 123857601272 737238112845712940348123 1934720871365475	1167220570724098896488008 1426349964088696127858578510648863253425

<b>Entrada</b>	<b>Saída</b>
6 0 104759 0 104801 0 105331 104743 0 104789 0 104987 0	0 0 0 0 0 0

<b>Entrada</b>	<b>Saída</b>
2 17546 92130935 737238112845712940348123 1934720871365475	1616529385510 1426349964088696127858578510648863253425

<b>Entrada</b>	<b>Saída</b>
2 0 92130935 737238112845712940348123 0	0 0



## 10 cálculo de áreas (+++)



(++)

Um grande amigo(a) seu(sua), dos tempos de colégio, está cursando Arquitetura na UFG e pediu auxílio para você para resolver o seguinte problema:

Ele precisa calcular a área, em metros quadrados, de diversas figuras planas:

C círculo – cujo raio é dado por  $R$ ;

E elipse – cujos raios maior e menor são, respectivamente,  $R$  e  $r$ ;

T triângulo – cujos lados são  $a$ ,  $b$  e  $c$  (nesta ordem);

Z trapézio – cujas bases maior e menor são, respectivamente,  $B$  e  $b$ , e a altura é  $H$  (nesta ordem).

Considera-se que vocês conhecem as “fórmulas matemáticas” de cálculo para as áreas destas figuras, mas você pensou numa solução mais sofisticada: elaborar um programa de computador C que seja capaz de receber as informações necessárias e retornar a área da figura.

**Observação:** Utilize  $\pi = 3,14159265$ .

### Entrada

A primeira linha da entrada contém um inteiro  $N$  ( $N \geq 1$ ) representando a quantidade de figuras planas para os quais seu programa deve calcular as áreas.

Cada uma das  $N$  linhas seguintes será composta por, primeiramente, um caractere que identifica qual é a figura e, em seguida, os parâmetros necessários para calcular sua área, na ordem anteriormente especificada e sempre separados por um único espaço em branco entre eles. Os parâmetros serão sempre números inteiros extraindo positivos.

### Saída

A saída consiste de  $N$  linhas, cada uma contendo a área da respectiva figura plana, com quatro casas decimais de precisão.

**Observação:** Considere que a resposta (área de cada figura plana), de acordo com seu(sua) amigo(a), precisa ter somente a parte inteira podendo, portanto, ser desprezada da parte decimal, desde que devidamente arredondada.

### Exemplos

<b>Entrada</b>	<b>Saída</b>
4 C 2 E 2 4 T 8 8 8 Z 7 3 4	13 25 28 20

<b>Entrada</b>	<b>Saída</b>
4 C 5 E 3 7 T 3 4 5 Z 7 10 4	79 66 6 34

<b>Entrada</b>	<b>Saída</b>
3 T 3 4 5 T 5 5 5 T 6 8 10	6 11 24

<b>Entrada</b>	<b>Saída</b>
3 E 5 5 E 4 8 E 1 2	79 101 6

<b>Entrada</b>	<b>Saída</b>
3 T 2 2 2 T 4 4 4 T 9 9 9	2 7 35



## 11 manipulação de matrizes 1 (++)



(++)

Um fundamental conceito abstrato da Matemática, extremamente utilizado em Computação, é o de *matriz*.

Uma matriz pode ser unidimensional (um vetor), bidimensional, tridimensional, etc.

Considerando apenas as matrizes bidimensionais  $A$ , de ordem  $m$  por  $n$ , que armazenam em cada uma de suas posições um número inteiro  $a_{i,j}$ , onde o índice  $i$  indica a linha e o índice  $j$  indica a coluna, com  $1 \leq i \leq m$  e  $1 \leq j \leq n$  e  $m, n \in \mathbb{N}^*$ , escreva um programa C que atenda às especificações indicadas a seguir.

### Entrada

A primeira linha da entrada contém os números naturais  $m$  e  $n$ , nesta ordem, separados por um único espaço em branco entre eles.

Cada uma das  $m$  linhas seguintes conterão os elementos localizados em cada uma das linhas da matriz  $A$ , separados entre si por um único espaço em branco. Sabemos, portanto, que cada uma destas linhas conterá  $n$  números inteiros.

A linha seguinte conterá um único caractere que indicará uma operação matricial: poderá ser o '+' (mais) para indicar a *adição* ou o 'x' (xis) para indicar a *multiplicação*.

Por fim, as últimas  $m$  linhas da entrada conterão os elementos localizados em cada uma das linhas da matriz  $B$ , separados entre si por um único espaço em branco. Sabemos, portanto, que cada uma destas linhas conterá  $n$  números inteiros.

**Observação:** Considere que  $1 \leq m, n \leq 10$  e que  $-50 \leq a_{i,j}, b_{i,j} \leq 50$ .

### Saída

A saída consistirá das linhas da matriz que corresponda à realização da operação  $A + B$  ou  $A \times B$ .

Lembre-se que há regras específicas para que a operação de multiplicação matricial possa ser realizada, bem como a maneira como esta se processa. Se não for possível realizá-la, o programa deverá emitir na saída uma linha com a mensagem: ERROR (grafada em letras maiúsculas).

## Exemplos

Entrada	Saída
2 3 1 2 3 4 5 6 + 6 5 4 3 2 1	7 7 7 7 7 7

Entrada	Saída
2 3 1 2 3 4 5 6 x 6 5 4 3 2 1	ERROR

Entrada	Saída
2 2 1 2 4 5 x 6 5 3 2	12 9 39 30

<b>Entrada</b>	<b>Saída</b>
<pre> 4 4 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 + 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 </pre>	<pre> 1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 </pre>

<b>Entrada</b>	<b>Saída</b>
<pre> 4 4 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 x 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 </pre>	<pre> 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 </pre>

$$A = \begin{bmatrix} 3 & 5 & 7 & 2 \\ 1 & 4 & 7 & 2 \\ 6 & 3 & 9 & 17 \\ 13 & 5 & 4 & 16 \end{bmatrix} \quad A^{-1} = \begin{bmatrix} 15 & 21 & 0 & 15 \\ 23 & 9 & 0 & 22 \\ 15 & 16 & 18 & 3 \\ 24 & 7 & 15 & 3 \end{bmatrix}$$

$$\det(A) = 21 \quad \det(A^{-1}) = 5$$

## 12 manipulação de matrizes 2 (++++)



(++++)

A partir do exercício anterior e, portanto, continuando com a proposta de considerar apenas as matrizes bidimensionais  $A$ , de ordem  $m$  por  $n$ , que armazenam em cada uma de suas posições um número inteiro  $a_{i,j}$ , onde o índice  $i$  indica a linha e o índice  $j$  indica a coluna, com  $1 \leq i \leq m$  e  $1 \leq j \leq n$  e  $m, n \in \mathbb{N}^*$ , escreva um programa C que atenda às especificações indicadas a seguir.

### Entrada

A primeira linha da entrada contém os números naturais  $m$  e  $n$ , nesta ordem, separados por um único espaço em branco entre eles.

Cada uma das  $m$  linhas seguintes conterão os elementos localizados em cada uma das linhas da matriz  $A$ , separados entre si por um único espaço em branco. Sabemos, portanto, que cada uma destas linhas conterá  $n$  números inteiros.

A linha seguinte conterá um único caractere que indicará uma operação matricial: poderá ser o 'I' (inversa) para indicar a operação de *inversão de matriz*, 'T' (tê) para indicar a *transposição de matriz* e, por fim, 'D' (dê) para indicar o *determinante* da matriz.

**Observação:** Considere que  $1 \leq m, n \leq 10$  e que  $-50 \leq a_{i,j}, b_{i,j} \leq 50$ .

### Saída

A saída consistirá das linhas da matriz que corresponda à realização da operação  $A^{-1}$  (matriz inversa de  $A$ ),  $A^t$  (matriz transposta de  $A$ ) ou  $\det(A)$  (determinante de  $A$ ).

Lembre-se que há regras específicas que regem a realização das operações de inversão de matriz e do cálculo de seu determinante. Se não for possível realizar a operação solicitada pelo usuário, o programa deverá emitir na saída uma linha com a mensagem: ERROR (grafada em letras maiúsculas).

### Exemplos

<b>Entrada</b>	<b>Saída</b>
2 3 1 2 3 4 5 6 T	1 4 2 5 3 6

<b>Entrada</b>	<b>Saída</b>
2 3 1 2 3 4 5 6 D	ERROR

<b>Entrada</b>	<b>Saída</b>
2 2 1 2 4 5 D	-3

<b>Entrada</b>	<b>Saída</b>
4 4 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 I	1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1

<b>Entrada</b>	<b>Saída</b>
4 4 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 D	1



## 13 Números de Fibonacci (++)



(++)

O matemático italiano Leonardo Fibonacci (1170-1250) foi de grande influência na Idade Média, sendo por muitos considerado como o maior deste período. Foi ele quem introduziu na Europa os *números arábicos* e descobriu uma curiosa sequência numérica que, por isso, foi posteriormente batizada de *Sequência de Fibonacci* e os números que a formam de *Números de Fibonacci*.

Aos 32 anos, Fibonacci publicou o livro *Liber Abaci* (ou seja, o *Livro do Ábaco* ou *Livro de Cálculo*), responsável pela disseminação dos números hindu-arábicos na Europa.

Como ele prestou grandes serviços à cidade de Pisa há nela uma estátua em sua homenagem, localizada na galeria ocidental do Camposanto (mostrada no cabeçalho desta questão).

Os *Números de Fibonacci* são definidos da seguinte maneira:

$$f_0 = 0$$

$$f_1 = 1$$

$$f_2 = 1$$

$$f_n = f_{n-1} + f_{n-2}, \text{ com } n \in \mathbb{N} \text{ e } n \geq 3$$

Escreva, em C, um programa que receba o valor de  $n$  conforme anteriormente definido,  $3 \leq n \leq 100$ , e escreva na saída o valor de  $f_n$  correspondente.

### Entrada

A primeira linha da entrada contém um número inteiro  $k$ ,  $1 \leq k \leq 10$ , que corresponde ao número de casos de teste que serão fornecidos nas linhas seguintes. Cada uma destas linhas conterá um valor específico

para  $n$ .

### Saída

Seu programa deve imprimir  $k$  linhas, cada uma contendo o valor calculado para o  $n$  correspondente na entrada.

### Exemplos

Entrada	Saída
4	2
3	3
4	5
5	8
6	

Entrada	Saída
7	13
7	21
8	34
9	55
10	89
11	144
12	233
13	

**Observação:** Lembre-se que um *Número de Fibonacci* pode ser extremamente grande. Por exemplo,  $f_{100} = 354224848179261915075$ . Portanto isto deve ser previsto no seu programa.

1296	2
648	2
324	2
162	2
81	3
27	3
9	3
3	3
1	

## 14 Fatoração de Números de Fibonacci (++++)



(++++)

Dando sequência ao estudo acerca dos *Números de Fibonacci*, o que se deseja agora é apresentar a *fatoração* de um certo número destes.

Você deverá escrever, novamente em C, um programa que receba o valor de  $k$  e, em sequência, um conjunto de valores  $n$  e imprima a fatoração de cada um dos  $f_n$  solicitados.

### Entrada

A primeira linha da entrada contém o número inteiro  $k$ ,  $1 \leq k \leq 10$ , que corresponde ao número de casos de teste que serão fornecidos nas linhas seguintes. Cada uma destas linhas conterá um valor para  $n$ ,  $1 \leq k \leq 100$ .

### Saída

Seu programa deve imprimir  $k$  linhas, cada uma contendo os fatores de  $f_n$ , apresentados em ordem estritamente crescente e separados por um espaço em branco.

### Exemplos

**Observação:** No último exemplo tem-se que  $f_{100} = 354224848179261915075$ . Sua fatoração é:  
 $3 \times 5 \times 5 \times 11 \times 41 \times 101 \times 151 \times 401 \times 3001 \times 570601$

<b>Entrada</b>	<b>Saída</b>
4 3 4 5 6	2 3 5 2 2 2

<b>Entrada</b>	<b>Saída</b>
7 7 8 9 10 11 12 13	13 3 7 2 17 5 11 89 2 2 2 2 3 3 233

<b>Entrada</b>	<b>Saída</b>
1 100	3 5 5 11 41 101 151 401 3001 570601