Algoritmos e Programação I: Lista de Exercícios 08-2 - Orientações para submissão no Moodle e BOCA. *

Faculdade de Computação
Universidade Federal de Mato Grosso do Sul
79070-900 Campo Grande, MS
`http://moodle.facom.ufms.br`

Após submeter no BOCA, não se esqueça de enviar o programa para o Moodle, usando o seguinte nome: pxxloginname.c, onde xx é o número do problema e o loginname é o seu login.

Compile o seu programa usando:

```
gcc -Wall -pedantic -std=c99 -o programa programa.c [-lm]
```

A flag `-lm` deve ser usada quando o programa incluir a biblioteca `math.h`.

Uma forma eficiente de testar se o seu programa está correto é gerando arquivos de entrada e saída e verificar a diferença entre eles. Isso pode ser feito da seguinte forma:

```
./programa < programa.in > programa.out
```

O conjunto de entrada deve ser digitado e salvo num arquivo `programa.in`. O modelo da saída deve ser digitado e salvo num arquivo `programa.sol`.

Verifique se a saída do seu programa `programa.out` é EXATAMENTE igual ao modelo de saída `programa.sol`. Isso pode ser feito com a utilização do comando `diff`, que verifica a diferença entre dois arquivos.

```
diff programa.out programa.sol
```

O seu programa só está correto se o resultado de `diff` for vazio (e se você fez todos os passos como indicado).

A solução dos exercícios deve seguir a metodologia descrita em sala:

a. Diálogo

b. Saída/Entrada (pós e pré)

c. Subdivisão

d. Abstrações

*Este material é para o uso exclusivo da disciplina de Algoritmos e Programação I da FACOM/UFMS e utiliza as referências bibliográficas da disciplina (B. Forouzan e R. Gilbert, A. B. Tucker et al. e S. Leestma e L. Nyhoff, Lambert et al., H. Farrer et al., K. B. Bruce et al., e Material Didático dos Profs. Edson Takashi Matsubara e Fábio Henrique Viduani Martinez.

e. Implementação

f. Teste

Na submissão ao BOCA não se esqueça de comentar todos os `printf`'s da entrada e que a saída não pode conter acentuação ou *ç*.

Exercícios

1. Suponha que os preços para os 15 modelos de automóveis do programa `vendauto.c` são as seguintes:

Modelo #	**Preço Modelo (R$)**
1	7.450,00
2	9.995,00
3	26.500,00
4	5.999,00
5	10.400,00
6	8.885,00
7	11.700,00
8	14.440,00
9	17.900,00
10	9.550,00
11	10.500,00
12	8.050,00
13	7.990,00
14	12.300,00
15	6.999,00

   Escreva um programa para ler esta lista de preços e a tabela de vendas dada em `vendauto.txt` e calcule o total de reais vendidos por cada vendedor e o total de vendas por todos os vendedores.

2. O programa de controle de estoque `estoque.c` simplesmente atualiza o matriz `JeansEmEstoque` com os valores das vendas fornecidos durante a execução do programa. Modifique o programa para permitir outras opções ao usuário como as seguintes:

   (a) Efetue uma busca no matriz `JeansEmEstoque` para determinar qual é o estoque atual de um determinado tipo de Jeans.

   (b) Imprima uma lista de pedido reposição de todos os nomes das marcas, estilos e tamanhos de jeans para o qual o estoque está abaixo de algum valor específico de pedido de reposição.

   (c) Faça o mesmo como em *b.*, mas neste caso imprima a lista dos jeans que estão encalhados para os quais o estoque está acima de um valor específico para encalhe.

3. Escreva um programa de controle de estoque semelhante ao `estoque.c`, mas para um vendedor de automóveis e utilize um vetor de cinco dimensões. O primeiro índice é a marca do carro `(Fiat, Ford, GM, Volkswagen)`, o segundo é o estilo `(DuasPortas, QuatroPortas, Caminhonete, Furgão)`, o terceiro é a cor `(azul, marrom, verde, vermelho, prata, amarelo)`, o quarto é o ano do veículo e o quinto é o código de vendas do veículo `(A, B, C)`.

4. O **código Morse** é o padrão de esquema de codificação que usa substituição e é semelhante ao esquema descrito anteriormente. As substituições usadas neste caso são mostradas na tabela abaixo. Escreva um programa para ler uma mensagem em texto normal ou em código morse e então codifique ou decodifique a mensagem.

A ·–	M ––	Y –·––
B –···	N –·	Z ––··
C –·–·	O –––	1 ·––––
D –··	P ·––·	2 ··–––
E ·	Q ––·–	3 ···––
F ··–·	R ·–·	4 ····–
G ––·	S ···	5 ·····
H ····	T –	6 –····
I ··	U ··–	7 ––···
J ·–––	V ···–	8 –––··
K –·–	W ·––	9 ––––·
L ·–··	X –··–	0 –––––

5. Um quadrado mágico é uma matriz $n \times n$ na qual cada inteiro $1, 2, 3, \ldots, n^2$ aparece exatamente uma vez e todas as somas das colunas, somas das linhas e somas das diagonais são iguais. Por exemplo, a tabela seguinte é uma quadrado mágico 5×5 na qual adicionando cada uma das linhas, colunas e diagonais é igual a 65.

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

A seguir damos os procedimento para a construção de um quadrado mágico $n \times n$ para um inteiro ímpar n. Coloque 1 no meio da primeira linha. Então após um inteiro k tenha sido colocado, movimente uma linha para cima e uma coluna para a direita e coloque o próximo inteiro $k + 1$, a menos que ocorra um dos seguintes:

i. Se um movimento leva você acima da primeira linha na j-ésima coluna, movimente então para a última linha na j-ésima coluna e coloque o inteiro nessa posição.

ii. Se um movimento leva você fora do lado direito do quadrado (última coluna) na i-ésima linha, coloque o inteiro na i-ésima linha no lado esquerdo do quadrado (primeira coluna).

iii. Se um movimento leva você a uma posição do quadrada já ocupada ou se o movimento leva você fora do quadrado no canto superior direito, coloque $k+1$ na posição imediatamente abaixo de k.

Escreva um programa para construir um quadrado mágico $n \times n$ para qualquer valor ímpar de n.

6. O famoso matemático G. H. Hardy uma vez mencionou ao jovem e brilhante matemático Indiano Ramanujan que o número do táxi que ele havia chegado era um número sem nenhuma característica especial. Ramanujan imediatamente respondeu que não, pois o número era muito interessante visto que ele era o menor número inteiro

que podia ser escrito como a soma de dois cubos (isto é, escrito da forma $x^3 + y^3$, com x e y inteiros) em duas formas diferentes. Escreva um programa para encontrar o número do táxi em que Hardy viajou.

7. Um **grafo direcionado** ou **digrafo** consiste de um conjunto de vértices e um conjunto de arestas direcionadas ligando alguns destes vértices. Por exemplo, a Figura 1 ilustra um grafo direcionado tendo cinco vértices numerados 1, 2, 3, 4 e 5 e sete arestas ligando os vértices 1 ao 2, 1 ao 4, 1 ao 5, 3 ao 1, 3 a ele mesmo, 4 ao 3, e 5 ao 1:

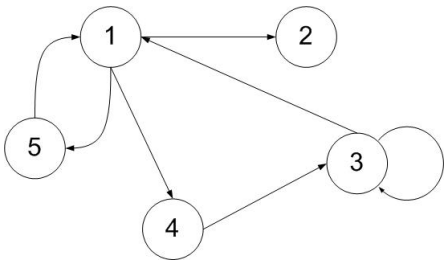


Figura 1: Uma matriz e sua decomposição em fatias.

Um grafo direcionado tendo n vértices pode ser representado por sua **matriz de adjacências** A, a qual é uma matriz $n \times n$, com a entrada na i-ésima linha e j-ésima coluna 1 se existe uma aresta direcionada do vértice i para o vértice j, 0 caso contrário. A matriz de adjacências para o grafo precedente é

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Se A é a matriz de adjacências para um grafo dirigido, a entrada na i-ésima linha e j-ésima coluna de A^k dá o número de maneiras que o vértice j pode ser atingido a partir do vértice i através de k arestas. Escreva um programa para ler o número de vértices de um grafo dirigido e uma coleção de pares ordenados de vértices representando as arestas direcionados; construa a matriz de adjacências; e então encontre o número de maneiras que cada vértice pode ser atingido por qualquer outro vértice através de k arestas para algum valor de k.

8. O **jogo da Vida**, inventado pelo matemático John H. Conway, foi motivado pela tentativa de modelar vida em uma sociedade de organismos. Considere um matriz retangular de células, cada uma das quais contém um organismo. Se a matriz é considerada como sendo prolongada indefinidamente em ambas as direções, então cada célula tem exatamente oito vizinhos, as oito células ao redor dela. Nascimentos e mortes ocorrem de acordo com as seguintes regras:

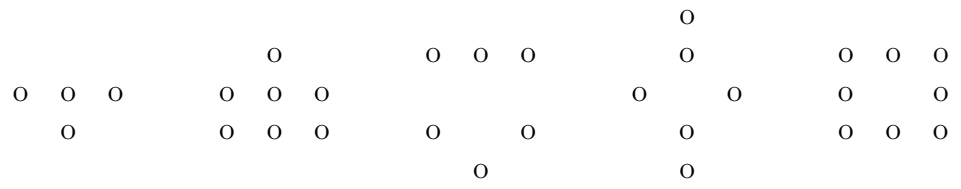
   i. Um organismo nasce em qualquer célula vazia que tenha exatamente três vizinhos vivos.

   ii. Um organismo morre de solidão se ele tem menos que dois vizinhos vivos.

   iii. Um organismos morre de sufocamento se ele tem mais que três vizinhos.

   iv. Todos os outros organismos sobrevivem para a próxima geração.

Para ilustrar, as figuras abaixo mostram as primeiras cinco gerações para uma dada configuração de organismos:



Escreva um programa para jogar o jogo da Vida e pesquisar padrões produzidos por várias configurações iniciais. Algumas configurações desaparecem rapidamente; outras se repetem após um determinado número de gerações; outras mudam a forma e tamanho e se movem através da matriz; e ainda outros produzem colônias que se separam da sociedade e navegam pelo espaço.

9. O **jogo de Nim** é jogado por dois jogadores. Geralmente existem três pilhas de objetos, e na vez da cada jogador, é permitido pegar qualquer número de objetos (pelo menos um) de uma pilha. O jogador pegando o último objeto perde. Escreva um programa que permita o usuário jogar contra o computador. Você pode ter um programa em que o computador jogue de forma perfeita, ou você pode projetar o seu programa para *ensinar* o computador. Uma forma do computador *aprender* é o de atribuir um valor para todo movimento possível, baseado na experiência obtida no decorrer dos jogos. O valor de cada movimento possível é armazenado em algum vetor; inicialmente, cada valor é 0. O valor de cada movimento em uma sequência de movimentos vencedora é aumentada por 1, e aqueles em uma sequência perdedora são diminuídos por 1. A cada estágio, o computador seleciona o melhor movimento possível (aquele tendo o maior valor).

10. O **jogo da Forca** é jogado por dois jogadores. Uma pessoa seleciona uma palavra e outra pessoa tenta adivinhar a palavra através da tentativa de letras individuais. Escreva um programa para jogar o jogo da Forca. Você pode armazenar uma lista de palavras em uma matriz ou arquivo e ter uma função para selecionar aleatoriamente uma palavra para o usuário adivinhar.

11. Escreva um programa que permita um usuário jogar o **jogo da Velha** contra o computador.

12. (alternada.c) [fhvm] Seja A uma matriz de dimensão $m \times n$. Uma **submatriz** W da matriz A que contém as linhas de i e j e as colunas de k a l de A é denotada por $W = A[i, j; k, l]$.

    Uma matriz de números inteiros A é dita ser uma **matriz alternada em** (i, j) se pode ser subdividida nas submatrizes $U = A[0, i; 0, j]$, $V = A[0, i; j + 1, n - 1]$, $X = A[i+1, m-1; 0, j]$ e $Y = A[i+1, m-1; j+1, n-1]$. Ademais, $U[i, j] \neq 0$ e $Y[i, j] \neq 0$, para todo i, j, e $V = X = 0$. Por exemplo,

$$\left(\begin{array}{rr|rrr} 1 & 3 & 0 & 0 & 0 \\ -1 & \mathbf{7} & 0 & 0 & 0 \\ \hline 0 & 0 & 4 & 1 & 2 \\ 0 & 0 & 3 & 6 & 1 \end{array}\right)$$

    é uma matriz alternada em $(1, 1)$.

    (a) Escreva uma função com a seguinte interface:

```
int altern(int m, int n, int A[MAX][MAX], int i, int j)
```

que receba dois números inteiros positivos m, n, uma matriz A de números inteiros de dimensão $m \times n$ e dois números inteiros i e j, e devolva 1 caso a matriz seja alternada em (i, j) e 0 em caso contrário.

(b) Escreva um programa que receba um número inteiro $k > 0$ que representa a quantidade de casos de teste. Para cada caso de teste, receba dois números inteiros m e n, com $1 \leq m, n \leq 100$, e uma matriz A de números inteiros de dimensão $m \times n$, e verifique se a matriz é alternada ou não-alternada. Em caso positivo, imprima os índices da linha e coluna do elemento para o qual a matriz é alternada. Caso contrário, imprima $-1 - 1$.

Exemplo de entrada:

```
alternada.in
2

4 5
1 3 0 0 0
-1 7 0 0 0
0 0 4 1 2
0 0 3 6 1

3 3
1 0 0
0 3 -4
2 1 0
```

Exemplo de saída:

```
alternada.in
1 1
-1 -1
```

13. (camada.c) [fhvm] A **camada** k de uma matriz quadrada A de dimensão n é um conjunto composto pelas seguintes células:

- **linha superior:** células $A(k, j)$, com $k \leq j \leq n - 1 - k$;
- **coluna esquerda:** células $A(i, k)$, com $k \leq i \leq n - 1 - k$;
- **linha inferior:** células $A(n - 1 - k, j)$, com $k \leq j \leq n - 1 - k$;
- **coluna direita:** células $A(i, n - 1 - k)$, com $k \leq i \leq n - 1 - k$.

Observe que, apesar da definição das linhas e colunas, não há repetição de células em uma camada. é mais fácil visualizar a camada k de uma matriz A na figura 2.

Observe que uma matriz quadrada A de dimensão n tem exatamente $\lceil n/2 \rceil$ camadas.

Dizemos que uma matriz quadrada A de dimensão n é **camada-ordenada** se a soma dos elementos de uma camada k é maior ou igual a soma dos elementos da camada $k + 1$, para todo k.

A

	0	1	2	3	4	5
0	1	2	5	1	−2	1
1	6	1	2	1	3	2
2	0	5	1	−2	0	3
3	1	2	5	3	3	1
4	−1	4	0	5	3	4
5	2	2	1	2	0	0

Figura 2: Uma matriz $A_{6\times 6}$ com a camada 1 destacada. Observe que a soma dos elementos da camada 0 é 31, da camada 1 é 29 e da camada 2 é 7. Dessa forma, a matriz A é camada-ordenada.

Por exemplo, as matrizes $\begin{pmatrix} 1 & 2 & 5 & 4 & 3 \\ 2 & 5 & 1 & 3 & 4 \\ 5 & 4 & 3 & 1 & 2 \\ 3 & 1 & 4 & 2 & 5 \\ 4 & 3 & 2 & 5 & 1 \end{pmatrix}$ e $\begin{pmatrix} 10 & 2 & -5 & 14 \\ 12 & 1 & 1 & -2 \\ 5 & -4 & 3 & 11 \\ 3 & 9 & 2 & 17 \end{pmatrix}$ são matrizes camadas-ordenadas, mas a matriz $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ não é camada-ordenada.

(a) Escreva uma função com a seguinte interface:

```
int soma_camada(int n, int A[MAX][MAX], int k)
```

que receba um número inteiro $n > 0$, uma matriz quadrada de números inteiros A de dimensão n e um índice k, e devolva a soma dos elementos da camada k de A.

(b) Escreva um programa que receba um número inteiro $t > 0$ que representa a quantidade de casos de teste. Para cada caso de teste, receba um número inteiro n, com $1 \leq n \leq 100$, e uma matriz quadrada de números inteiros de dimensão n, e verifique se é camada-ordenada ou não. Em caso positivo, imprima `C-O`. Caso contrário, imprima `N`.

Exemplo de entrada:

```
camada.in
2

4
10 2 -5 14
12 1 1 -2
5 -4 3 11
3 9 2 17
3
1 1 1
```

```
1 8 1
1 1 0
```

Exemplo de saída:

```
camada.sol
C-O
N
```

14. (diagonal.c) [fhvm] Uma **diagonal** de uma matriz A de dimensões $m \times n$ é uma sequência das seguintes células da matriz:

    - $(0, j), (1, j+1), (2, j+2), \ldots, (k, j+k)$, para algum k tal que $k = m-1$ ou $j+k = n-1$; ou
    - $(i, 0), (i+1, 1), (i+2, 2), \ldots, (i+k, k)$, para algum k tal que $i+k = m-1$ ou $k = n-1$.

    Chamamos a diagonal que inicia em $(0, j)$ de **diagonal** $0, j$. Do mesmo modo, chamamos a diagonal que inicia em $(i, 0)$ de **diagonal** $i, 0$.

    Chamamos uma diagonal $0, j$ (ou $i, 0$) de **diagonal par** se j (ou i) é par. Se j (ou i) é ímpar, então a diagonal é chamada de **diagonal ímpar**.

    A figura 3 ilustra diagonais de uma matriz.

A

	0	1	2	3	4	5
0	2	3	6	5	1	6
1	1	2	1	5	4	2
2	3	1	4	2	3	1
3	2	4	2	3	8	1

Figura 3: Diagonais de uma matriz $A_{4\times6}$. Uma diagonal par $0, 2$ e uma diagonal ímpar $1, 0$.

Dizemos, ainda, que uma matriz A de números inteiros positivos de dimensões $m \times n$ é **diagonal-alternada** se a soma dos elementos de suas diagonais pares é ímpar e a soma dos elementos de suas diagonais ímpares é par. Por exemplo, as matrizes abaixo são diagonais-alternadas:

$$\begin{pmatrix} 1 & 2 & 1 & 2 & 3 \\ 6 & 1 & 2 & 3 & 2 \\ 3 & 4 & 4 & 4 & 1 \\ 2 & 5 & 2 & 2 & 0 \\ 5 & 2 & 1 & 2 & 1 \end{pmatrix} \quad \text{e} \quad \begin{pmatrix} 1 & 3 & 3 \\ 4 & 2 & 3 \end{pmatrix}.$$

Mas a matriz abaixo não é diagonal-alternada:

$$\begin{pmatrix} 1 & 1 \\ 4 & 3 \\ 2 & 4 \\ 0 & 1 \\ 5 & 2 \end{pmatrix}.$$

(a) Escreva uma função com a seguinte interface:

```
int soma_diag(int m, int n, int A[MAX][MAX], int i, int j)
```

que receba dois números inteiros positivos m, n, uma matriz A de números inteiros positivos de dimensão $m \times n$ e um par de números inteiros i, j, e devolva a soma dos elementos da diagonal i, j. Observe que, como se trata de uma diagonal, $i = 0$ ou $j = 0$.

(b) Escreva um programa que receba um número inteiro $k > 0$ que representa a quantidade de casos de teste. Para cada caso de teste, receba um par de números inteiros m, n, com $1 \leq m, n \leq 100$, e uma matriz de números inteiros positivos de dimensão $m \times n$, e verifique se a matriz é diagonal-alternada ou não. Em caso positivo, imprima `D-A`. Caso contrário, imprima `N`.

Exemplo de entrada:

```
diagonal.in
2

4 6
2 3 6 5 1 6
1 2 1 5 4 2
3 1 4 2 3 1
2 4 2 3 8 1

3 3
1 1 1
1 1 1
1 1 2
```

Exemplo de saída:

```
diagonal.sol
D-A
N
```

15. (fatias.c) [fhvm] Seja A uma matriz quadrada de números inteiros de ordem n, com $1 \leq n \leq 100$. Podemos subdividir a matriz A em 6 componentes: (i) diagonal principal; (ii) diagonal secundária; (iii) fatia superior; (iv) fatia inferior; (v) fatia esquerda; e (vi) fatia direita. Na verdade, as diagonais principal e secundária delimitam essas fatias, como podemos ver na figura 4. Chamamos essa decomposição de **decomposição de A em fatias**.

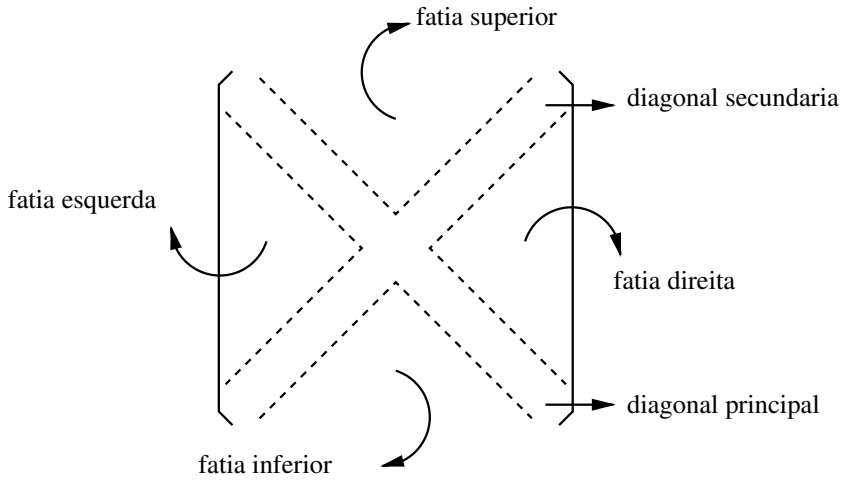


Figura 4: Uma matriz e sua decomposição em fatias.

Dada uma matriz quadrada A de números inteiros de ordem $n \geq 1$ de números inteiros e sua decomposição em fatias, dizemos que A é um **X de ordem** n, ou simplesmente um X_n, se A satisfaz as seguintes condições:

- a diagonal principal é composta apenas por números 1;
- a diagonal secundária é composta apenas por números 1;
- a fatia superior contém números inteiros cuja soma é maior ou igual à soma dos números inteiros que compõem a fatia inferior;
- a fatia esquerda contém números inteiros cuja soma é maior ou igual à soma dos números inteiros que compõem a fatia direita.

Exemplo:

$$\text{A matriz } A = \begin{pmatrix} \mathbf{1} & 4 & -3 & 8 & \mathbf{1} \\ 7 & \mathbf{1} & 7 & \mathbf{1} & 2 \\ 9 & -6 & \mathbf{1} & 3 & -2 \\ 9 & \mathbf{1} & 0 & \mathbf{1} & 3 \\ \mathbf{1} & 2 & 0 & -1 & \mathbf{1} \end{pmatrix} \text{ é um } X_5.$$

(a) Escreva uma função com a seguinte interface:

```
int diagonais(int n, int A[MAX][MAX])
```

que receba um número inteiro $n > 0$ e uma matriz quadrada A de dimensão n, e verifique se as diagonais principal e secundária de A contém apenas o número 1, devolvendo 1 em caso positivo e 0 em caso negativo.

(b) Escreva uma função com a seguinte interface:

```
int sup_inf(int n, int A[MAX][MAX])
```

que receba um número inteiro $n > 0$ e uma matriz quadrada A de dimensão n, e verifique se a soma dos elementos da fatia superior é maior ou igual a soma dos elementos da fatia inferior, devolvendo 1 em caso positivo e 0 em caso negativo.

(c) Escreva uma função com a seguinte interface:

```
int esq_dir(int n, int A[MAX][MAX])
```

que receba um número inteiro $n > 0$ e uma matriz quadrada A de dimensão n, e verifique se a soma dos elementos da fatia esquerda é maior ou igual a soma dos elementos da fatia direita, devolvendo 1 em caso positivo e 0 em caso negativo.

(d) Escreva um programa que receba um número inteiro $k > 0$ que representa a quantidade de casos de teste. Para cada caso de teste, receba um número inteiro n, com $1 \leq n \leq 100$, e uma matriz quadrada de dimensão n, e verifique se é um X_n. Em caso positivo, imprima `X`n, onde n é a dimensão da matriz de entrada. Caso contrário, imprima `X0`.

Exemplo de entrada:

```
fatias.in
2

4
1 2 2 1
2 1 1 3
3 1 1 4
1 2 3 1

5
1 4 -3 8 1
7 1 7 1 2
9 -6 1 3 -2
9 1 0 1 3
1 2 0 -1 1
```

Exemplo de saída:

```
fatias.sol
X0
X5
```

16. `(alfabetica.c)` [fhvm] Uma matriz de caracteres A, com m linhas e n colunas, é dita **linha-alfabética** se os caracteres de cada linha estão dispostos em ordem crescente. Isto é, se

- $a_{i,j} < a_{i,j+1}$ para todo par i, j, com $1 \leq i \leq m$ e $1 \leq j < n$.

Do mesmo modo, dizemos que a matriz A é **coluna-alfabética** se os caracteres de cada coluna estão dispostos em ordem crescente. Ou seja, se

- $a_{i,j} < a_{i+1,j}$ para todo par i, j, com $1 \leq i < m$ e $1 \leq j \leq n$.

Dizemos ainda que a matriz A é **toda-alfabética** se os caracteres de cada linha estão dispostos em ordem crescente e de cada coluna também estão dispostos em ordem crescente. Formalmente, a matriz $A_{m \times n}$ é toda-alfabética se:

- $a_{i,j} < a_{i,j+1}$ para todo par i, j, com $1 \leq i \leq m$ e $1 \leq j < n$, e
- $a_{i,j} < a_{i+1,j}$ para todo par i, j, com $1 \leq i < m$ e $1 \leq j \leq n$.

(a) Escreva uma função com a seguinte interface:

```
\int linha(char u[MAX+1])
```

que receba uma cadeia de caracteres u e devolva 1 se os caracteres que ocorrem nas células de u estão dispostos em ordem crescente. Caso contrário, devolva 0.

(b) Escreva uma função com a seguinte interface:

```
int coluna(int m, char A[MAX][MAX+1], int j)
```

que receba um número inteiro $m > 0$, uma matriz A com m linhas onde cada linha é uma cadeia de caracteres, e um índice j, e devolva 1 se os caracteres que ocorrem nas células da coluna j de A estão dispostos em ordem crescente. Caso contrário, devolva 0.

(c) Escreva um programa que receba um número inteiro $k > 0$ que representa a quantidade de casos de teste. Para cada caso de teste, receba um par de números inteiros m, n, com $1 \leq m, n \leq 100$, e uma matriz A de dimensão $m \times n$, onde cada linha é uma cadeia de caracteres, e verifique se a matriz é linha-alfabética, coluna-alfabética, toda-alfabética ou não-alfabética. Considere que os caracteres fornecidos na entrada são todos minúsculos. Na saída, imprima `L`, `C`, `T` ou `N` para uma matriz linha-alfabética, coluna-alfabética, toda-alfabética ou não-alfabética, respectivamente.

Exemplo de entrada:

```
alfabetica.in
4

3 3
ahl
dep
bcq

3 4
acbe
fdhk
gijl

3 4
ajux
hkvy
ilwz

3 3
ahl
dim
fen
```

Exemplo de saída:

```
alfabetica.sol
L
```

```
C
T
N
```

17. (`idempontente.c`) [fhvm] Uma matriz A de números inteiros de dimensão $n \times n$ é chamada **idempotente** se e somente se $A^2 = A \times A = A$. Por exemplo, a matriz

$$\begin{pmatrix} 4 & -6 \\ 2 & -3 \end{pmatrix}$$

é uma matriz idempotente já que

$$A^2 = \begin{pmatrix} 4 & -6 \\ 2 & -3 \end{pmatrix} \times \begin{pmatrix} 4 & -6 \\ 2 & -3 \end{pmatrix} = \begin{pmatrix} 4 & -6 \\ 2 & -3 \end{pmatrix} = A\,.$$

(a) Escreva uma função com a seguinte interface:

```
void mult\_mat(int n, int A[MAX][MAX], int B[MAX][MAX], int C[MAX][MAX])
```

que receba um número inteiro $n > 0$ e duas matrizes quadradas A e B de números inteiros, de dimensão n, e compute a matriz $C = A \times B$, também de dimensão n.

(b) Escreva uma função com a seguinte interface:

```
int iguais(int n, int A[MAX][MAX], int B[MAX][MAX])
```

que receba um número inteiro $n > 0$ e duas matrizes A e B de dimensão n, e devolva 1 se $A = B$ e 0, caso contrário.

(c) Escreva um programa que receba um número inteiro $k > 0$ que indica o número de casos de teste. Para cada caso de teste, receba um número inteiro n, com $1 \leq n \leq 100$, e uma matriz A de números inteiros de dimensão n, e verifique, usando as funções dos itens (a) e (b), se A é idempotente. Em caso positivo, imprima `I`. Caso contrário, imprima `N`.

Exemplo de entrada:

```
alfabetica.in
2

2
4 -6
2 -3

3
1 0 1
3 5 1
0 2 3
```

Exemplo de saída:

```
alfabetica.sol
I
N
```

18. (`latino.c`) [fhvm] Dizemos que uma matriz $A_{n\times n}$ é um quadrado latino de ordem n se em cada linha e em cada coluna aparecem todos os inteiros $1, 2, 3, \ldots, n$, ou seja, cada linha e coluna é permutação dos inteiros $1, 2, \ldots, n$.

    Exemplo:
    A matriz

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \end{pmatrix}$$

    é um quadrado latino de ordem 4.

    (a) Escreva uma função com a seguinte interface:

```
int linha(int n, int A[MAX][MAX], int i)
```

    que receba um número inteiro $n > 0$, uma matriz quadrada $A_{n\times n}$ de números inteiros e um índice i, e verifique se a linha i de A contém todos os números inteiros de 1 a n, devolvendo 1 em caso positivo e 0 caso contrário.

    (b) Escreva uma função com a seguinte interface:

```
int coluna(int n, int A[MAX][MAX], int j)
```

    que receba um número inteiro $n > 0$, uma matriz quadrada $A_{n\times n}$ de números inteiros e um índice j, e verifique se a coluna j de A contém todos os números inteiros de 1 a n, devolvendo 1 em caso positivo e 0 caso contrário.

    (c) Escreva um programa que receba um número inteiro $k > 0$, que representa a quantidade de casos de teste; para cada caso de teste, receba um número inteiro $n > 0$ e uma matriz quadrada de dimensão n e verifique se a matriz é um quadrado latino. Em caso positivo imprima `L` e em caso negativo imprima `N`.

    Exemplo de entrada:

```
latino.in
2

2
1 2
2 1

3
1 2 3
2 3 1
3 1 4
```

    Exemplo de saída:

```
latino.sol
L
N
```

19. `(lecker.c)` [fhvm] Dizemos que uma sequência de números inteiros, sem elementos repetidos, é uma **sequência lecker** se tem apenas um elemento que é maior que seus vizinhos. Por exemplo,

    - 2 5 10 46 25 12 7 é lecker, pois 46 é o único elemento que é maior que seus vizinhos, que são 10 e 25;
    - 13 5 4 2 3 0 −3 −14 não é lecker, porque 13 é maior que 5 (5 é o único vizinho do 13) e 3 é maior que 2 e 0 (2 e 0 são os vizinhos do 3);
    - 6 7 8 9 10 é lecker, pois apenas 10 é maior que seus vizinhos (9 é o único vizinho do 10);
    - 7 6 é lecker, pois apenas 7 é maior que seus vizinhos (6 é o único vizinho de 7).

    Uma matriz é uma **matriz lecker** se suas linhas e colunas são sequências lecker.

    (a) Escreva uma função com a seguinte interface:

    ```
    int linha(int n, int u[MAX])
    ```

    que receba um número inteiro $n > 0$ e um vetor u de números inteiros com n elementos, e devolva 1 se u contém uma sequência lecker e 0 em caso contrário.

    (b) Escreva uma função com a seguinte interface:

    ```
    int coluna(int m, int A[MAX][MAX], int j)
    ```

    que receba um número inteiro $m > 0$, uma matriz A de números inteiros com m linhas e um índice j, e devolva 1 se a coluna j de A é uma sequência lecker e 0 em caso contrário.

    (c) Escreva um programa que receba um número inteiro $k > 0$ que representa a quantidade de casos de teste. Para cada caso de teste, receba um par de números inteiros m e n, com $1 \leq m, n \leq 100$, e uma matriz A de números inteiros de dimensão $m \times n$, e verifique se a matriz é lecker. Na saída, imprima `L` ou `N` para uma matriz lecker ou não-lecker, respectivamente.

    Exemplo de entrada:

    ```
    lecker.in
    2

    3 7
    2 5 10 46 25 12 7
    6 7 8 9 11 13 14
    -1 0 3 4 -2 -5 -6

    5 5
    4 5 8 7 6
    10 21 13 12 9
    11 12 14 15 16
    20 19 18 17 3
    21 2 1 0 22
    ```

Exemplo de saída:

```
lecker.sol
L
N
```

20. (`maximos.c`) [fhvm] Dada uma matriz de números reais A com m linhas e n colunas, dizemos que uma posição (i, j) da matriz é um **l-máximo** se o elemento $A(i, j)$ é o máximo da linha i, ou seja, se

$$A(i, j) \geq A(i, k), \text{ para todo } k = 0, \ldots, n-1 \,.$$

De forma análoga, podemos definir que uma posição da matriz é um **c-máximo** se o elemento correspondente à posição for o elemento máximo da coluna.

Exemplo:

Considere a matriz abaixo:

$$A = \begin{pmatrix} 13.0 & 6.0 & 4.0 & 28.0 & 7.5 & 1.05 \\ 4.0 & 0.0 & 0.3 & -13.0 & 12.3 & 128.1 \\ -1.0 & 8.0 & 2.0 & -1.9 & 12.0 & 2.0 \\ -2.0 & 7.0 & 5.0 & 0.0 & 4.0 & 12.0 \end{pmatrix} .$$

Entã,

- o elemento 5.0 na posição (3,2) é um c-máximo;
- o elemento 12.0 na posição (2,4) é um l-máximo;
- o elemento 28.0 na posição (0,3) é um lc-máximo; e
- o elemento 4.0 na posição (1,0) não é um máximo.

(a) Escreva uma função com a seguinte interface:

```
float l\_maximo(int n, float A[MAX][MAX], int i)
```

que receba um número inteiro n, uma matriz A de números reais com n colunas e um índice i, e devolva o l-máximo da linha i da matriz A.

(b) Escreva uma função com a seguinte interface:

```
float c\_maximo(int m, float A[MAX][MAX], int j)
```

que receba um número inteiro m, uma matriz A de números reais com m linhas e um índice j, e devolva o c-máximo da coluna j da matriz A.

(c) Escreva um programa que receba um número inteiro $k > 0$ que representa a quantidade de casos de teste. Para cada caso de teste, receba um par de números inteiros m e n, com $1 \leq m, n \leq 100$, e uma matriz de números reais A de dimensão $m \times n$; além disso, receba uma lista de pares de números inteiros i e j, com $0 \leq i \leq m-1$ e $0 \leq j \leq n-1$, e verifique, para cada par de índices, se o elemento $A(i, j)$ é um l-máximo, um c-máximo, um lc-máximo ou não é máximo, imprimindo `C`, `L`, `LC` ou `N`, respectivamente. O último par de índices informado é `-1 -1`.

Exemplo de entrada:

```
maximos.in
1

4 6
13.0 6.0 4.0 28.0 7.5 1.05
4.0 0.0 0.3 -13.0 12.3 128.1
-1.0 8.0 2.0 -1.9 12.0 2.0
-2.0 7.0 5.0 0.0 4.0 12.0
3 2
2 4
0 3
1 0
-1 -1
```

Exemplo de saída:

```
maximos.sol
C
L
LC
N
```

21. (`ordlinhas.c`) [fhvm] Uma matriz de dígitos decimais A, de dimensão $m \times n$, pode ser usada para representar números inteiros de n dígitos, estendendo a precisão dos números inteiros que podem ser representados no computador. Podemos imaginar, por exemplo, que cada célula da matriz armazena um dígito e que cada linha representa um número inteiro de n dígitos.

    Exemplo: A matriz

    $$\begin{pmatrix} 1 & 5 & 1 & 0 & 1 & 2 \\ 3 & 4 & 5 & 6 & 9 & 9 \\ 0 & 0 & 0 & 1 & 0 & 7 \\ 6 & 1 & 8 & 7 & 8 & 4 \end{pmatrix}$$

    armazena o número 151.012 na primeira linha, o número 345.699 na segunda, o número 107 na terceira e o número 618.784 na última.

    (a) Escreva uma função com a seguinte interface:

    ```
    void troca(int n, int u[MAX], int v[MAX])
    ```

    que receba um número inteiro n e dois vetores u e v de números inteiros, de dimensão n, e troca os seus conteúdos coordenada por coordenada.

    (b) Escreva uma função com a seguinte com interface:

    ```
    int verifica\_maior(int n, int A[MAX][MAX], int i, int j)
    ```

    que receba um número inteiro $n > 0$, uma matriz A de números inteiros com n colunas e um par de índices i, j, e devolve 1 se o número representado na linha i é maior que o número da linha j. Caso contrário, a função devolve 0.

(c) Escreva uma função com a seguinte com interface:

```
void ordena(int m, int n, int A[MAX][MAX])
```

que receba números inteiros m e n e uma matriz de números inteiros A de dimensão $m \times n$ e, usando as funções dos itens anteriores, coloca os números representados nas linhas da matriz em ordem crescente, modificando assim a matriz A.

(d) Escreva um programa que receba um número inteiro $k > 0$ indicando a quantidade de casos de teste. Para cada caso de teste, receba dois números inteiros m e n, com $1 \leq m, n \leq 100$, e uma matriz A de números inteiros de dimensão $m \times n$, onde cada célula contém um dígito de 0 a 9, e, usando as funções dos itens anteriores, coloca as linhas da matriz A em ordem crescente, mostrando essa matriz resultante na saída. Entre duas matrizes resultantes, imprima uma linha em branco.

Exemplo de entrada:

```
ordlinhas.in
2

3 5
0 1 2 9 7
9 6 5 4 4
0 0 0 1 2

4 6
1 5 1 0 1 2
3 4 5 6 9 9
0 0 0 1 0 7
6 1 8 7 8 4
```

Exemplo de saída:

```
ordlinhas.sol
0 0 0 1 2
0 1 2 9 7
9 6 5 4 4

0 0 0 1 0 7
1 5 1 0 1 2
3 4 5 6 9 9
6 1 8 7 8 4
```

Observação: na saída, na apresentação de uma matriz resultante, sempre informe um elemento e um espaço em uma linha; isso significa que o último elemento da linha vem sempre sucedido de um espaço e, então, de uma mudança de linha; todas as linhas seguem essa regra; imprima também uma mudança de linha entre duas matrizes que são soluções.

22. (`ordmatriz.c`) [fhvm] Seja A uma matriz de números inteiros. O **sucessor** do elemento $a_{i,j}$ da matriz é (i) o elemento seguinte na mesma linha, se $a_{i,j}$ não é o último elemento da linha, ou (ii) o primeiro elemento da próxima linha da matriz, caso $a_{i,j}$ seja o último elemento da linha.

    Por exemplo, na matriz abaixo, o sucessor de 13 é o elemento 5, o sucessor de 5 é 17, o sucessor de 17 é 0, o sucessor de 0 é 9, ..., o sucessor de 3 é 15, ..., e o elemento 7 não tem sucessor:

$$\begin{pmatrix} 13 & 5 & 17 & 0 \\ 9 & -1 & 6 & 3 \\ 15 & 10 & 20 & 7 \end{pmatrix}.$$

    Dizemos que uma matriz está **ordenada** se cada elemento desta matriz é menor ou igual ao seu sucessor.

    Exemplo de matriz ordenada:

$$\begin{pmatrix} -1 & 0 & 3 & 5 \\ 6 & 7 & 9 & 10 \\ 13 & 15 & 17 & 20 \end{pmatrix}.$$

    Exemplo de matrizes que não estão ordenadas:

$$\begin{pmatrix} -1 & 0 & 3 & 5 \\ 13 & 15 & 17 & 20 \\ 6 & 7 & 9 & 10 \end{pmatrix} \quad \begin{pmatrix} -1 & 0 & 5 & 3 \\ 6 & 7 & 9 & 10 \\ 13 & 15 & 17 & 20 \end{pmatrix} \quad \begin{pmatrix} -1 & 0 & 3 & 5 \\ 6 & 7 & 9 & 13 \\ 10 & 15 & 17 & 20 \end{pmatrix}.$$

    (a) Escreva uma função com a seguinte interface:

```
int ordenada(int m, int n, int A[MAX][MAX], int *l, int *c)
```

    que receba números inteiros positivos m, n e uma matriz A de números inteiros de dimensão $m \times n$, e devolva 1 se a matriz está ordenada e 0 caso contrário. Além disso, no caso da matriz não estar ordenada, a função deve devolver os índices l, c da linha e coluna de um elemento $a_{l,c}$ que seja maior que o seu sucessor.

    Exemplos:

    Para $m = 3, n = 4$ e a matriz

$$A = \begin{pmatrix} -1 & 0 & 3 & 5 \\ 6 & 7 & 9 & 10 \\ 13 & 15 & 17 & 20 \end{pmatrix}$$

    a função devolve 1.

    Para $m = 3, n = 4$ e a matriz

$$A = \begin{pmatrix} -1 & 0 & 3 & 5 \\ 13 & 15 & 17 & 20 \\ 6 & 7 & 9 & 10 \end{pmatrix}$$

    o valor de l devolvido pela função deve ser 1, o valor de c deve ser 3 e a função devolve 0.

Para $m = 3, n = 4$ e a matriz

$$A = \begin{pmatrix} -1 & 0 & 3 & 5 \\ 13 & 17 & 15 & 20 \\ 6 & 7 & 9 & 10 \end{pmatrix}$$

os valores de l e c devolvidos pela função podem ser 1 e 1 ou 1 e 3, e a função devolve 0.

(b) Escreva uma função com a seguinte interface:

```
void troca(int m, int n, int A[MAX][MAX], int l, int c)
```

que receba um par de números inteiros positivos m, n, uma matriz A de números inteiros de dimensão $m \times n$ e dois números inteiros l e c, e troque o valor do elemento $a_{l,c}$ com o valor de seu sucessor. Você pode supor que o elemento $a_{l,c}$ tem sucessor.

Exemplo:

Para $m = 3, n = 4, l = 0, c = 3$ e matriz

$$A = \begin{pmatrix} -1 & 0 & 3 & 5 \\ 13 & 17 & 15 & 20 \\ 6 & 7 & 9 & 10 \end{pmatrix}$$

a matriz determinada pela função deve ser

$$\begin{pmatrix} -1 & 0 & 3 & 13 \\ 5 & 17 & 15 & 20 \\ 6 & 7 & 9 & 10 \end{pmatrix}.$$

(c) Escreva um programa que receba um número inteiro $k > 0$ que representa a quantidade de casos de teste. Para cada caso de teste, receba dois números inteiros m e n, com $1 \leq m, n \leq 100$, e uma matriz A de números inteiros de dimensão $m \times n$, ordene a matriz A e imprima a matriz ordenada. Use as funções dos itens (a) e (b) para ir trocando elementos da matriz até obter uma matriz ordenada.

Exemplo de entrada:

```
ordmatriz.in
2

2 2
3 -1
7 0

3 3
2 5 0
3 2 6
1 4 2
```

Exemplo de saída:

```
ordmatriz.sol
-1 0
3 7

0 1 2
2 2 3
4 5 6
```

Observação: na saída, na apresentação de uma matriz resultante, sempre informe um elemento e um espaço em uma linha; isso significa que o último elemento da linha vem sempre sucedido de um espaço e, então, de uma mudança de linha; todas as linhas seguem essa regra; imprima também uma mudança de linha entre duas matrizes que são soluções.

23. `(potencia.c)` [fhvm] A k-ésima potência de uma matriz quadrada $A_{n\times n}$, denotada por A^k, é a matriz obtida a partir da matriz A da seguinte forma:

$$A^k = I_n \times \overbrace{A \times A \times \ldots \times A}^{k},$$

onde I_n é a matriz identidade de ordem n. Para facilitar a computação de A^k, podemos usar a seguinte fórmula:

$$A^k = \begin{cases} I_n\,, & \text{se } k = 0\,, \\ A^{k-1} \times A\,, & \text{se } k > 0\,. \end{cases}$$

Escreva um programa que leia um número inteiro $t > 0$, que representa a quantidade de casos de teste; para cada caso de teste, receba um número inteiro n, com $1 \leq n \leq 100$, um número inteiro $k > 0$ e uma matriz quadrada A de números inteiros de dimensão n, e compute e imprima A^k utilizando as funções abaixo.

a. A função `identidade` tem a seguinte:

```
void identidade(int n, int I[MAX][MAX])
```

e recebe um número inteiro $n > 0$ e uma matriz quadrada I de números inteiros de dimensão n e preencha I com os valores da matriz identidade de ordem n.

b. A função `mult_mat` tem a seguinte interface:

```
void mult_mat(int n, int C[MAX][MAX], int A[MAX][MAX],
              int B[MAX][MAX])
```

e recebe um número inteiro $n > 0$ e matrizes quadradas A, B e C de números inteiros, todas de dimensão n, e compute $C = A \times B$.

c. A função `copia` tem a seguinte interface:

```
void copia(int n, int A[MAX][MAX], int B[MAX][MAX])
```

e recebe um número inteiro $n > 0$ e as matrizes quadradas A e B de números inteiros de dimensão n e copie os elementos da matriz B na matriz A.

Entrada: A primeira linha contem um inteiro t indicando a quantidade de casos de teste, seguido de uma sequencia contendo a dimensão das matrizes, a potência a ser calculada e as matrizes.

```
potencia.in
2

2
1
1 2
3 4

3
3
1 0 1
0 1 0
1 0 1
```

Saída: A saída consiste da potência calculada da matriz de entrada.

```
potencia.sol
1 2
3 4

4 0 4
0 1 0
4 0 4
```

Observação: na saída, na apresentação de uma matriz resultante, sempre informe um elemento e um espaço em uma linha; isso significa que o último elemento da linha vem sempre sucedido de um espaço e, então, de uma mudança de linha; todas as linhas seguem essa regra; imprima também uma mudança de linha entre duas matrizes que são soluções.