

Competidor(a): \_\_\_\_\_

Número de inscrição: \_\_\_\_\_ (opcional)

*Este Caderno de Tarefas não pode ser levado para casa após a prova. Após a prova entregue este Caderno de Tarefas para seu professor guardar. Os professores poderão devolver os Cadernos de Tarefas aos competidores após o término do período de aplicação das provas (12 a 14 de Junho de 2024).*



Olimpíada Brasileira de Informática

OBI2023

Caderno de Tarefas

Modalidade Programação • Nível Sênior • Fase 1

12 a 14 de Junho de 2024

A PROVA TEM DURAÇÃO DE 2 horas

Promoção:



Sociedade Brasileira de Computação

Apoio:



Coordenação:



# Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 10 páginas (não contando a folha de rosto), numeradas de 1 a 10. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta ao *help* do ambiente de programação se este estiver disponível.
- As tarefas têm o mesmo valor na correção.
- A correção é automatizada, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa; em particular, seu programa não deve escrever frases como “Digite o dado de entrada:” ou similares.
- Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc.), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
- As tarefas **não** estão necessariamente ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
- Preste muita atenção no nome dos arquivos fonte indicados nas tarefas. Soluções na linguagem C devem ser arquivos com sufixo *.c*; soluções na linguagem C++ devem ser arquivos com sufixo *.cc* ou *.cpp*; soluções na linguagem Java devem ser arquivos com sufixo *.java* e a classe principal deve ter o mesmo nome do arquivo fonte; soluções na linguagem Python 3 devem ser arquivos com sufixo *.py*; e soluções na linguagem Javascript devem ter arquivos com sufixo *.js*.
- Na linguagem Java, **não** use o comando *package*, e note que o nome de sua classe principal deve usar somente letras minúsculas (o mesmo nome do arquivo indicado nas tarefas).
- Você pode submeter até 50 soluções para cada tarefa. A pontuação total de cada tarefa é a melhor pontuação entre todas as submissões. Se a tarefa tem sub-tarefas, para cada sub-tarefa é considerada a melhor pontuação entre todas as submissões.
- Não utilize arquivos para entrada ou saída. Todos os dados devem ser lidos da entrada padrão (normalmente é o teclado) e escritos na saída padrão (normalmente é a tela). Utilize as funções padrão para entrada e saída de dados:
  - em C: *scanf*, *getchar*, *printf*, *putchar*;
  - em C++: as mesmas de C ou os objetos *cout* e *cin*.
  - em Java: qualquer classe ou função padrão, como por exemplo *Scanner*, *BufferedReader*, *BufferedWriter* e *System.out.println*
  - em Python: *read*, *readline*, *readlines*, *input*, *print*, *write*
  - em Javascript: *scanf*, *printf*
- Procure resolver a tarefa de maneira eficiente. Na correção, eficiência também será levada em conta. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.

# Ogro

Nome do arquivo: `ogro.c`, `ogro.cpp`, `ogro.java`, `ogro.js` ou `ogro.py`

Ogro e Bicho-Papão têm fama de malvados, mas na verdade são amáveis, honestos e trabalhadores, além de vizinhos e amigos. O Bicho-Papão tem dificuldades em aprender aritmética e por isso o Ogro inventou uma brincadeira simples para auxiliar seu amigo: o Ogro inicia mostrando um certo número de dedos na sua mão esquerda (vamos chamar esse valor de  $E$ ) e um número de dedos diferente na mão direita (vamos chamar esse valor de  $D$ ). Então, Bicho-Papão deve falar o *resultado* da brincadeira, definido assim:

- se o número de dedos na mão esquerda é maior do que o número de dedos na mão direita (ou seja  $E > D$ ) então o resultado é a soma dos dois números (ou seja  $E + D$ );
- caso contrário, o resultado é o dobro da diferença entre o número de dedos na mão direita e o número de dedos na mão esquerda (ou seja,  $2 \times (D - E)$ ).

O problema é que o Ogro também não é lá muito bom em aritmética, e pediu sua ajuda para conferir se o Bicho-Papão falou a resposta correta.

Dados o número de dedos mostrados na mão esquerda ( $E$ ) e o número de dedos mostrados na mão direita ( $D$ ), escreva um programa para determinar a resposta da brincadeira.

## Entrada

A entrada é composta por duas linhas. A primeira linha contém um inteiro  $E$ , o número de dedos mostrados na mão esquerda. A segunda linha contém um inteiro  $D$ , o número de dedos mostrados na mão direita.

## Saída

Seu programa deve produzir uma única linha na saída, contendo um único número inteiro, o resultado da brincadeira.

## Restrições

- $0 \leq E \leq 5$
- $0 \leq D \leq 5$
- $E \neq D$

## Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (30 pontos):**  $E > D$ .
- **Subtarefa 3 (70 pontos):** Sem restrições adicionais.

Seu programa pode resolver corretamente todas ou algumas das subtarefas acima (*elas não precisam ser resolvidas em ordem*). Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por qualquer uma das suas submissões.

**Exemplos**

<b>Exemplo de entrada 1</b>  1 0	<b>Exemplo de saída 1</b>  1
<b>Exemplo de entrada 2</b>  2 5	<b>Exemplo de saída 2</b>  6

# Relógio

*Nome do arquivo:* `relogio.c`, `relogio.cpp`, `relogio.java`, `relogio.js` ou `relogio.py`

A Seleção de Basquete Campinense (SBC) está ansiosa para disputar a final do campeonato nacional universitário contra o Clube de Basquete Sergipano (CBS), o arquirrival da SBC. Entretanto, a comissão organizadora esteve sobrecarregada e acabou cometendo alguns erros logísticos, o que levou o jogo a ser adiado em exatamente  $T$  segundos. A comissão já está trabalhando dobrado e precisa da sua ajuda para reajustar o horário do jogo.

A sua tarefa é: dado o horário original de início do jogo e o tempo  $T$ , em segundos, em que o jogo foi adiado, determine o novo horário de início do jogo.

## Entrada

A entrada contém quatro linhas. As três primeiras linhas indicam o horário original de início do jogo: a primeira linha contém um inteiro  $H$ , que indica as horas. A segunda linha contém um inteiro  $M$ , que indica os minutos. A terceira linha contém um inteiro  $S$ , que indica os segundos. A quarta linha contém um único inteiro  $T$ , indicando em quantos segundos o jogo foi adiado.

Note que o horário é dado no formato de 24 horas, ou seja,  $H$  é um inteiro entre 0 e 23.

## Saída

Seu programa deve imprimir o novo horário de início do jogo, seguindo o mesmo formato de horário da entrada. Ou seja, seu programa deve imprimir três linhas, cada uma contendo um único inteiro:

- A primeira linha deve conter as horas do novo horário de início do jogo.
- A segunda linha deve conter os minutos do novo horário de início do jogo.
- A terceira linha deve conter os segundos do novo horário de início do jogo.

Observe que você não deve imprimir zeros à esquerda (veja o exemplo de saída 2).

## Restrições

- $0 \leq H \leq 23$
- $0 \leq M \leq 59$
- $0 \leq S \leq 59$
- $0 \leq T \leq 10^9$

## Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (60 pontos):**  $T = 1$ .

- **Subtarefa 3 (20 pontos):**  $T \leq 10^5$ .
- **Subtarefa 4 (20 pontos):** Sem restrições adicionais.

Seu programa pode resolver corretamente todas ou algumas das subtarefas acima (*elas não precisam ser resolvidas em ordem*). Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por qualquer uma das suas submissões.

## Exemplos

<b>Exemplo de entrada 1</b>  3 14 15 1	<b>Exemplo de saída 1</b>  3 14 16
<b>Exemplo de entrada 2</b>  11 59 59 1	<b>Exemplo de saída 2</b>  12 0 0
<b>Exemplo de entrada 3</b>  20 24 0 2024	<b>Exemplo de saída 3</b>  20 57 44
<b>Exemplo de entrada 4</b>  1 0 1 99101	<b>Exemplo de saída 4</b>  4 31 42

# Placas de Carro

Nome do arquivo: `placas.c`, `placas.cpp`, `placas.java`, `placas.js` ou `placas.py`

As placas usadas nos carros em circulação no Brasil possuem dois padrões com formatos diferentes: algumas placas estão no antigo padrão Brasileiro, enquanto outras estão no novo padrão Mercosul.

O antigo padrão Brasileiro é sempre formado por 8 caracteres:

- os três primeiros caracteres são letras maiúsculas (de A a Z);
- o quarto caractere é um hífen (-);
- os últimos quatro caracteres são dígitos (de 0 a 9).

Por exemplo, `OBI-2024` é uma placa válida no antigo padrão Brasileiro.

O novo padrão Mercosul, por sua vez, é sempre formado por 7 caracteres:

- os três primeiros caracteres são letras maiúsculas;
- o quarto caractere é um dígito;
- o quinto caractere é uma letra maiúscula;
- os últimos dois caracteres são dígitos.

Assim, `OBI2P24` é uma placa válida no novo padrão Mercosul.

Há também um grande contingente de carros em situação irregular – carros com placas falsificadas que não estão nem no antigo padrão Brasileiro, nem no novo padrão Mercosul. Por exemplo, um carro com a placa `OBI-24` está em situação irregular, pois a placa não é válida em nenhum dos dois padrões.

O Departamento Nacional de Trânsito identificou que seus funcionários gastam muito tempo verificando manualmente quais placas estão em qual padrão e quais são falsificadas. Por isso, eles pediram sua ajuda para automatizar o processo: dada uma placa formada por uma sequência de letras maiúsculas, dígitos e hífen, determine se a placa está no antigo padrão Brasileiro, está no novo padrão Mercosul, ou é uma placa falsificada.

## Entrada

A entrada é composta de uma única linha, contendo uma sequência de caracteres representando a placa a ser analisada.

## Saída

Seu programa deverá imprimir uma linha contendo um único número inteiro:

- 1, se a placa está no antigo padrão Brasileiro;
- 2, se a placa está no novo padrão Mercosul;
- 0, se a placa é falsificada.

## Restrições

- A placa possui entre 6 e 10 caracteres.
- Cada caractere da placa é uma letra maiúscula (de A a Z), um dígito (de 0 a 9) ou um hífen (o caractere -).

## Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (40 pontos):** É garantido que a placa está ou no antigo padrão Brasileiro, ou no novo padrão Mercosul (ou seja, ela não é falsificada).
- **Subtarefa 3 (60 pontos):** Sem restrições adicionais.

Seu programa pode resolver corretamente todas ou algumas das subtarefas acima (*elas não precisam ser resolvidas em ordem*). Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por qualquer uma das suas submissões.

## Exemplos

<b>Exemplo de entrada 1</b>  OBI-2024	<b>Exemplo de saída 1</b>  1
<b>Exemplo de entrada 2</b>  OBI2P24	<b>Exemplo de saída 2</b>  2
<b>Exemplo de entrada 3</b>  OBI-24	<b>Exemplo de saída 3</b>  0
<b>Exemplo de entrada 4</b>  XYZ-1234	<b>Exemplo de saída 4</b>  1
<b>Exemplo de entrada 5</b>  A1B2C3D4E5	<b>Exemplo de saída 5</b>  0



# Jogo da Vida

Nome do arquivo: `jogo.c`, `jogo.cpp`, `jogo.java`, `jogo.js` ou `jogo.py`

O Jogo da Vida de Conway é um processo de simulação (conhecido como *autômato celular*) criado pelo matemático britânico John Conway para reproduzir, por meio de uma matriz, processos de mudança em grupos de seres vivos. As regras do jogo indicam como a matriz é modificada a cada passo. Os valores da matriz em um determinado passo são coletivamente chamados de *estado* do jogo.

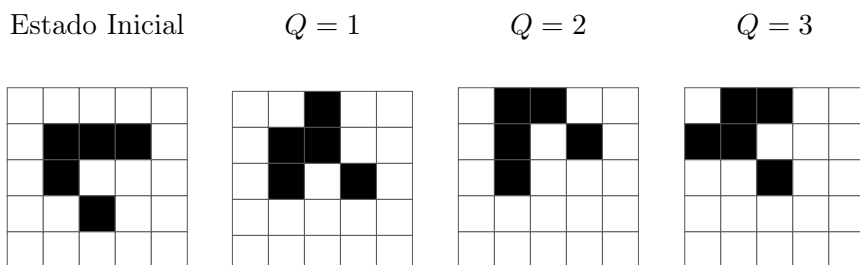
Mais especificamente, o jogo acontece em uma matriz quadrada  $N \times N$  (ou seja, com  $N$  linhas e  $N$  colunas) no qual cada célula está viva (representada pelo número 1) ou morta (representada pelo número 0). Para simular o próximo estado do autômato, para cada célula calculamos o seu número de vizinhos vivos (duas células são consideradas vizinhas se elas são adjacentes diagonalmente, horizontalmente ou verticalmente – ou seja, uma célula pode ter até 8 vizinhas), e decidimos se a célula estará viva ou morta no próximo estado de acordo com as seguintes regras:

- se uma célula morta possui exatamente três vizinhas vivas, ela vira uma célula viva;
- se uma célula morta possui uma quantidade de vizinhas vivas diferente de três, ela continua morta;
- se uma célula viva possui duas ou três vizinhas vivas, ela continua viva;
- se uma célula viva possui menos que duas vizinhas vivas, ela morre;
- se uma célula viva possui mais que três vizinhas vivas, ela morre.

Toda célula fora da matriz é considerada morta, ou seja, células fora da matriz nunca afetam a quantidade de vizinhos vivos de alguma célula. Observe que as regras são aplicadas em todas as células simultaneamente, uma vez a cada passo.

Dada uma matriz que representa o estado inicial do jogo e um inteiro positivo  $Q$ , sua tarefa é determinar o  $Q$ -ésimo estado do jogo de acordo com as regras descritas acima, ou seja, o valor de cada célula da matriz após  $Q$  passos do jogo.

A figura abaixo mostra um exemplo de jogo em uma matriz  $5 \times 5$  e seus estados para diferentes valores de  $Q$ . Células vivas são representadas com a cor preta e células mortas são representadas com a cor branca.



## Entrada

A primeira linha contém dois números inteiros,  $N$  e  $Q$ , representando, respectivamente, o número de linhas/colunas da matriz e o número de passos a serem simulados.

As próximas  $N$  linhas contêm  $N$  caracteres cada. O  $j$ -ésimo caractere da  $i$ -ésima linha representa o estado inicial da célula na linha  $i$  e coluna  $j$ . Caso o caractere seja '0', a célula naquela posição inicia o jogo morta; caso o caractere seja '1', a célula inicia o jogo viva.

## Saída

O seu programa deverá imprimir  $N$  linhas, cada uma contendo  $N$  caracteres. Na  $i$ -ésima linha, o  $j$ -ésimo caractere deve representar o  $Q$ -ésimo estado da célula na linha  $i$  e coluna  $j$ . Caso a célula esteja morta, o caractere deve ser '0'; se ela estiver viva, o caractere deve ser '1'.

## Restrições

- $1 \leq N \leq 50$
- $1 \leq Q \leq 100$

## Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (30 pontos):**  $Q = 1$ .
- **Subtarefa 3 (70 pontos):** Sem restrições adicionais.

Seu programa pode resolver corretamente todas ou algumas das subtarefas acima (*elas não precisam ser resolvidas em ordem*). Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por qualquer uma das suas submissões.

## Exemplos

Exemplo de entrada 1	Exemplo de saída 1
5 3	01100
00000	11000
01110	00100
01000	00000
00100	00000
00000	

<p><b>Exemplo de entrada 2</b></p> <pre> 15 1 000010000010000 000010000010000 000011000110000 000000000000000 111001101100111 001010101010100 000011000110000 000000000000000 000011000110000 001010101010100 111001101100111 000000000000000 000011000110000 000010000010000 000010000010000 </pre>	<p><b>Exemplo de saída 2</b></p> <pre> 000000000000000 000110000011000 000011000110000 010010101010010 011101101101110 001010101010100 000111000111000 000000000000000 000111000111000 001010101010100 011101101101110 010010101010010 000011000110000 000110000011000 000000000000000 </pre>
<p><b>Exemplo de entrada 3</b></p> <pre> 15 3 000010000010000 000010000010000 000011000110000 000000000000000 111001101100111 001010101010100 000011000110000 000000000000000 000011000110000 001010101010100 111001101100111 000000000000000 000011000110000 000010000010000 000010000010000 </pre>	<p><b>Exemplo de saída 3</b></p> <pre> 000010000010000 000010000010000 000011000110000 000000000000000 111001101100111 001010101010100 000011000110000 000000000000000 000011000110000 001010101010100 111001101100111 000000000000000 000011000110000 000010000010000 000010000010000 </pre>