

## Problema B - Giro da Fortuna

Este é um **problema de interação**.

Ao contrário dos outros problemas em que deves fazer leitura de dados e escrita do output, neste problema deves interagir com o avaliador através da implementação de uma função e da interação com as funções fornecidas.

Um dos desafios mais apaixonantes no programa de televisão ONI (Onde Nada é Impossível) é a roleta da sorte, também apelidada de Giro da Fortuna pelos fãs mais fervorosos. A roleta está dividida em  $N$  setores circulares idênticos, numerados de 0 a  $N - 1$  no sentido dos ponteiros do relógio. Em cada setor foi colocada uma lâmpada, que pode estar ligada ou desligada. O estado inicial de cada lâmpada é para ti, concorrente, desconhecido: como a roleta está virada de costas para ti, apenas o apresentador e o público conseguem ver o estado das lâmpadas em qualquer momento.



O teu objetivo enquanto concorrente é conseguir **desligar** todas as lâmpadas em simultâneo. As regras do Giro da Fortuna determinam duas operações que podes fazer:

1. **flip( $i$ )**. Escolhes um inteiro  $i$  ( $0 \leq i \leq N - 1$ ) e pedes ao apresentador para inverter o estado da lâmpada que se encontra atualmente no setor  $i$ : se a lâmpada estiver ligada, então passa a estar desligada; senão passa a estar ligada.
2. **check()**. Perguntas ao apresentador se atualmente todas as lâmpadas estão desligadas. Se estiverem, ganhas de imediato o desafio; senão, o apresentador gira aleatoriamente a roleta e o desafio continua. Quando a roleta é girada, os números dos setores que as lâmpadas ocupam são deslocados ciclicamente para a direita.

Existe um pequeno porém: tens direito a, no máximo,  $Q$  perguntas do tipo **check()**; se fizeres mais serás desqualificado(a). Não existe limite de pedidos do tipo **flip( $i$ )**.

Serás capaz de vencer o desafio e passar à próxima fase do programa?

## Ficheiros para Download

Podes começar por descarregar os ficheiros correspondentes à tua linguagem (ou um [arquivo zip](#) contendo tudo):

Linguagem	Ficheiro a implementar	Avaliador	Outros Ficheiros	Input exemplo
C++	<a href="#">resolver.cpp</a>	<a href="#">avaliador.cpp</a>	<a href="#">avaliador.h</a>	<a href="#">input.txt</a>

Nota que a implementação do avaliador a usar nos testes oficiais será diferente.

## Implementação

Deves submeter um único ficheiro que implementa uma função:

- A função `find(N, Q)`, que recebe um inteiro  $N$  (que representa o número de setores circulares da roleta) e um inteiro  $Q$  (que representa o número máximo de vezes que podes fazer perguntas do tipo `check()`).

Para isso deves usar o ficheiro `resolver.cpp` que descarregaste, colocando no interior das funções o teu código. Podes acrescentar outras funções, mas devem ficar todas neste ficheiro que é o único que deves submeter.

### Funções a implementar:

C++: `void find(int N, int Q)`

A tua função deve invocar as seguintes funções:

- A função `flip(pos)`, que recebe um inteiro  $pos$  (o número do setor circular) e não devolve nenhum valor.
- A função `check()`, que não recebe nenhum valor e devolve um booleano indicando se todas as lâmpadas se encontram atualmente desligadas. Quando é executada esta função, se a resposta devolvida for `false` a roleta é girada arbitrariamente.

Nota o seguinte:

- Os valores de  $pos$  devem estar compreendidos entre 0 e  $N - 1$ . Se alguma chamada não respeitar estas condições, o teu código terá o resultado de **Wrong Answer**.
- Se excederes  $Q$  perguntas do tipo `check()` (ou seja, se fizeres mais do que  $Q$  chamadas à função `check()`), o teu código terá o resultado de **Wrong Answer**.
- O avaliador demora cerca de 20 segundos a responder a 70 000 chamadas da função `check` (verifica os limites de tempo e restrições para perceber a relevância desta nota).

### Funções do avaliador:

C++: `void flip(int pos)`

C++: `bool check()`

A tua função não deve ler nem escrever para os canais de entrada/saída padrão.

## Exemplo

Vamos supor que há  $N = 4$  setores circulares e que há um máximo de  $Q = 20$  perguntas. Suponhamos também o seguinte estado inicial das lâmpadas nos setores  $0, 1, \dots, N - 1$  (0 denota desligada, 1 denota ligada): 1 0 0 1.

Uma possível execução seria a seguinte (com a ordem das chamadas a ser de cima para baixo):

Invocação	Resultado	Descrição	Estado das lâmpadas
–	–	Início	1 0 0 1
flip(1)	–	Lâmpada no setor 1 ligada	1 1 0 1
check()	false	A roleta gira 2 unidades para a direita	0 1 1 1
flip(2)	–	Lâmpada no setor 2 desligada	0 1 0 1
flip(3)	–	Lâmpada no setor 3 desligada	0 1 0 0
check()	false	A roleta gira 1 unidade para a direita	0 0 1 0
flip(2)	–	Lâmpada no setor 2 desligada	0 0 0 0
check()	true	Vitória	0 0 0 0

## Restrições

São garantidos os seguintes limites em todos os casos de teste que irão ser colocados ao programa:

- $4 \leq N \leq 16$  Número de setores circulares da roleta
- $1 \leq Q$  Número máximo permitido de perguntas do tipo `check()`

## Sumário de subtarefas

Os casos de teste do problema estão organizados em quatro grupos com restrições adicionais diferentes:

Grupo	Número de Pontos	Restrições adicionais
1	10	A roleta não é girada após cada pergunta, $N = 16$ e $Q \geq 70\,000$
2	30	$N = 4$ , $Q \geq 20$
3	30	$N = 8$ , $Q \geq 300$
4	30	$N = 16$ , $Q \geq 70\,000$

## Testes no vosso computador

É disponibilizado um avaliador exemplo em cada linguagem (`avaliador.cpp`) que pode ser utilizado para testar a tua submissão. Está ainda disponível um ficheiro auxiliar (`avaliador.h`).

Este avaliador não corresponde ao utilizado pelo sistema de avaliação.

Este avaliador começa por receber como input um inteiro  $N$  e um inteiro  $Q$ , correspondendo, respetivamente, ao número de setores circulares e ao número máximo de perguntas. Segue-se uma linha com  $N$  inteiros, representando o estado de cada lâmpada inicialmente, onde cada um é 0 (desligada) ou 1 (ligada). Finalmente segue-se uma outra linha com  $Q$  inteiros, sendo cada um entre 1 e  $N$ , representando cada uma das rotações da roleta (por exemplo, se o quarto valor de rotação for 3, então a roleta será girada 3 unidades para a direita no caso em que o concorrente não vence ao fazer a quarta pergunta).

O avaliador irá automaticamente invocar a função `find(N, Q)` por ti implementada. Disponibilizamos um ficheiro de teste:

- [input.txt](#) que contém o caso de exemplo referido acima.

Um exemplo de teste na tua máquina (supondo que tens os compiladores oficiais instalados) seria o seguinte:

Linguagem	Compilar	Executar com o exemplo
C++	<code>g++ -Wall -std=gnu++14 -O2 avaliador.cpp resolver.cpp</code>	<code>./a.out &lt; input.txt</code>

#### Organização



#### Alto Patrocínio

Com o Alto Patrocínio  
de Sua Excelência



O Presidente da República



REPÚBLICA  
PORTUGUESA

GABINETE DO MINISTRO DA EDUCAÇÃO

#### Patrocinadores



Fundação  
para a Ciência  
e a Tecnologia



FUNDAÇÃO  
CALOUSTE  
GULBENKIAN



NTT DATA

#### Apoios

