

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 (14 e 15 de agosto de 2025).*



Olimpíada Brasileira de Informática

OBI2025

Caderno de Tarefas

Modalidade Programação • Nível Júnior • Fase 2

14 e 15 de agosto de 2025

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 7 páginas (não contando a folha de rosto), numeradas de 1 a 7. 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.

# Arara!

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

Consideradas por muitos como símbolos da fauna brasileira, as araras são aves que se destacam pelas suas penas coloridas, animação e inteligência. Assim, a chegada de novas araras ao zoológico de São Paulo é sempre muito aguardada pelo público.

Esta semana, o zoológico deseja fazer um evento para apresentar suas  $N$  araras ao público. Na região do zoológico onde o evento vai acontecer, existem  $M$  gaiolas alinhadas, numeradas de 1 a  $M$  da esquerda para a direita. Cada gaiola pode abrigar uma única arara ou ficar vazia.

Os funcionários gostariam de distribuir as  $N$  araras entre as  $M$  gaiolas para o evento. No entanto, as araras tem a tendência de expressar emoções gritando *Arara!* de maneira inesperada. Quando uma arara grita, outras araras que estejam em gaiolas muito próximas podem se assustar com o barulho e também começar a gritar, o que por sua vez pode assustar outras araras e assim por diante.

Felizmente, pesquisadores do zoológico descobriram que uma arara se assusta com o grito de outra arara somente se existem menos do que quatro gaiolas entre elas (desconsiderando suas próprias gaiolas). Por exemplo, o grito de uma arara na gaiola 8 assustaria uma arara na gaiola 6 (existe apenas uma gaiola entre elas) ou uma arara na gaiola 12 (existem apenas três gaiolas entre elas), mas não assustaria uma arara na gaiola 3 (existem quatro gaiolas entre elas) ou na gaiola 16 (existem sete gaiolas entre elas).

O zoológico decidiu que o evento pode ser realizado se é possível distribuir as  $N$  araras entre as  $M$  gaiolas de modo que nenhum grito *Arara!* possa assustar outras araras, ou seja, de modo que existam pelo menos quatro gaiolas vazias entre quaisquer duas araras. Sua tarefa é determinar se o evento pode ser realizado.

## Entrada

A entrada possui uma única linha de entrada contendo dois inteiros  $N$  e  $M$  indicando, respectivamente, o número de araras e o número de gaiolas.

## Saída

Seu programa deverá imprimir uma única linha contendo um único caractere. Caso seja possível distribuir as araras entre as gaiolas para realizar o evento, imprima o caractere **S** (a letra S maiúscula). Caso seja impossível, imprima o caractere **N** (a letra N maiúscula).

## Restrições

É garantido que todo caso de teste satisfaz as restrições abaixo.

- $1 \leq N \leq 1000$
- $1 \leq M \leq 1000$

## Informações sobre a pontuação

A tarefa vale 100 pontos.

**Exemplos**

<b>Exemplo de entrada 1</b> 2 10	<b>Exemplo de saída 1</b> S
<b>Exemplo de entrada 2</b> 3 10	<b>Exemplo de saída 2</b> N
<b>Exemplo de entrada 3</b> 3 11	<b>Exemplo de saída 3</b> S

# Placar do Jogo

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

Durante uma viagem para a Bolívia, Paulo e Camila encontraram um jogo de futebol de mesa. Como bons brasileiros, os dois já haviam jogado esse jogo no Brasil, e pretendiam simplesmente continuar o passeio. No entanto, um terrível impasse surgiu: Paulo dizia que o jogo se chamava Pebolim, mas Camila insistia que o nome era Totó! Diante desse importante dilema e sem a possibilidade de entendimento mútuo, a disputa foi inevitável.

A partida foi longa e intensa, de forma que as crianças perderam as contas de quantos gols fizeram. Prevendo que isso aconteceria, eles pediram a Afonso – um amigo que chamava o jogo de Pacau e, portanto, era um juiz imparcial – que marcasse o placar. Ao fim do jogo, Afonso havia anotado não só as quantidades  $P$  e  $C$  de gols que Paulo e Camila fizeram, respectivamente, como também os momentos (em minutos) em que cada gol foi marcado. Vale ressaltar que os gols ocorreram em minutos **distintos** e foram anotados em **ordem cronológica**.

Após o fim da partida, Paulo e Camila nem lembravam o porquê da disputa, mas estavam curiosos a respeito da evolução do placar ao longo do jogo. Por exemplo, suponha que Paulo fez dois gols, nos minutos 24 e 38, enquanto Camila fez três gols, nos minutos 1, 21 e 63. Os placares que existiram durante a partida são:

- $0 \times 0$ : início da partida.
- $0 \times 1$ : Camila marca um gol no minuto 1.
- $0 \times 2$ : Camila marca um gol no minuto 21.
- $1 \times 2$ : Paulo marca um gol no minuto 24.
- $2 \times 2$ : Paulo marca um gol no minuto 38.
- $2 \times 3$ : Camila marca um gol no minuto 63.

Perceba que o placar sempre é mostrado como a quantidade de gols de Paulo, seguida da quantidade de gols de Camila, nesta ordem.

Sua tarefa é: dadas as anotações de Afonso, ou seja, as quantidades de gols e os momentos em que os gols foram marcados para cada criança, determine a lista completa e cronologicamente ordenada de todos os placares que existiram durante a partida.

## Entrada

A primeira linha da entrada contém um inteiro  $P$ , a quantidade de gols marcados por Paulo, seguido de  $P$  valores,  $p_1, p_2, \dots, p_P$ , os momentos (em minutos) dos gols de Paulo, em ordem cronológica.

A segunda linha contém um inteiro  $C$ , a quantidade de gols marcados por Camila, seguido de  $C$  valores,  $c_1, c_2, \dots, c_C$ , os momentos (em minutos) dos gols de Camila, em ordem cronológica.

## Saída

Seu programa deverá imprimir  $P + C + 1$  linhas, uma para cada placar que existiu em algum momento da partida, em ordem cronológica. Para cada placar, imprima dois números inteiros, a quantidade de gols de Paulo e a quantidade de gols de Camila, nesta ordem.

## Restrições

É garantido que todo caso de teste satisfaz as restrições abaixo.

- $0 \leq P, C \leq 50$
- $1 \leq p_1 < p_2 < \dots < p_P \leq 100$
- $1 \leq c_1 < c_2 < \dots < c_C \leq 100$
- $p_i \neq c_j$  para todo  $1 \leq i \leq P$  e  $1 \leq j \leq C$  (ou seja, os minutos em que houve um gol são todos distintos)

## 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 (18 pontos):**  $P = 1$  e  $C = 1$ .
- **Subtarefa 3 (16 pontos):**  $P = 0$ .
- **Subtarefa 4 (24 pontos):** Houve um gol a cada minuto da partida. Ou seja, houve um gol no minuto 1, um gol no minuto 2, e assim por diante até o minuto  $P + C$ .
- **Subtarefa 5 (42 pontos):** Sem restrições adicionais.

## Exemplos

<p><b>Exemplo de entrada 1</b></p> <p>2 24 38 3 1 21 63</p>	<p><b>Exemplo de saída 1</b></p> <p>0 0 0 1 0 2 1 2 2 2 2 3</p>
<p><b>Exemplo de entrada 2</b></p> <p>0 2 60 70</p>	<p><b>Exemplo de saída 2</b></p> <p>0 0 0 1 0 2</p>
<p><b>Exemplo de entrada 3</b></p> <p>0 0</p>	<p><b>Exemplo de saída 3</b></p> <p>0 0</p>

# Feirinha de Artesanato

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

A tradicional feira anual de artesanatos da sua cidade está chegando. O dono de uma das barracas mais populares da feira pediu a sua ajuda para registrar o lucro da barraca ao fim do dia.

Existem dois tipos de objetos que podem ser vendidos na barraca, o tipo 1 e o tipo 2. No início do dia, você registra todo o estoque atual da barraca, que contém  $N$  objetos numerados de 1 a  $N$ . O  $i$ -ésimo objeto possui tipo  $t_i$  e preço  $p_i$  (em reais). Observe que a barraca pode possuir mais de um objeto do mesmo tipo em estoque, e **não** é garantido que ambos os tipos estão em estoque.

Durante o dia,  $C$  clientes vão visitar a barraca, um de cada vez. Todo cliente vai comprar no máximo um objeto, pagando à barraca o preço dele. Cada cliente pode ser decidido ou indeciso:

- Um cliente *decidido* sabe qual tipo de objeto ele deseja comprar. Ao visitar a barraca, um cliente decidido compra o objeto do tipo desejado que possui o menor preço. Caso existam mais de um objeto com o tipo desejado e preço mínimo, ele compra qualquer um destes objetos, mas somente um. Caso não exista nenhum objeto com o tipo desejado disponível, o cliente decidido vai embora sem comprar nada.
- Um cliente *indeciso* se importa mais com o preço do objeto do que com o tipo, usando o tipo do objeto apenas como critério de desempate. Mais especificamente, ao visitar a barraca, um cliente indeciso compra o objeto que possui menor preço entre todos os objetos disponíveis. Caso existam vários objetos com preço mínimo, ele compra o objeto cujo tipo é mínimo. O cliente indeciso só vai embora sem comprar nada caso não existam mais objetos disponíveis.

Vale ressaltar que cada um dos  $N$  objetos só pode ser comprado uma vez, e um objeto que é comprado é removido do estoque.

Sua tarefa é calcular o valor total que a barraca arrecadou com vendas após as visitas dos  $C$  clientes.

## Entrada

A primeira linha da entrada possui um único inteiro  $N$ , a quantidade de objetos em estoque no início do dia.

A segunda linha da entrada possui  $N$  inteiros  $t_1, t_2, \dots, t_N$ , os tipos dos  $N$  objetos.

A terceira linha da entrada possui  $N$  inteiros  $p_1, p_2, \dots, p_N$ , os preços em reais dos  $N$  objetos, na mesma ordem da linha anterior.

A quarta linha da entrada possui um único inteiro  $C$ , o número de clientes que vão visitar a barraca.

A quinta linha da entrada possui  $C$  inteiros  $u_1, u_2, \dots, u_C$  e descrevem os clientes **na ordem em que visitaram a barraca**. Mais especificamente:

- Se o  $j$ -ésimo cliente é decidido,  $u_j$  é o tipo de objeto desejado.
- Se o  $j$ -ésimo cliente é indeciso,  $u_j = 0$ .

## Saída

Seu programa deverá imprimir uma única linha contendo um único inteiro, o total em reais que a barraca recebeu ao longo do dia.

## Restrições

É garantido que todo caso de teste satisfaz as restrições abaixo.

- $1 \leq N \leq 100\,000$
- $1 \leq C \leq 100\,000$
- $1 \leq t_i \leq 2$  para  $1 \leq i \leq N$
- $1 \leq p_i \leq 10\,000$  para  $1 \leq i \leq N$
- $0 \leq u_j \leq 2$  para  $1 \leq j \leq C$

## 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 (17 pontos):**  $N \leq 100$  e  $C \leq 100$ .
- **Subtarefa 3 (14 pontos):** Todos os clientes são indecisos.
- **Subtarefa 4 (21 pontos):** Todos os clientes são decididos.
- **Subtarefa 5 (15 pontos):** Todos os objetos possuem preço 1.
- **Subtarefa 6 (33 pontos):** Sem restrições adicionais.

## Exemplos

Exemplo de entrada 1	Exemplo de saída 1
<pre>7 2 1 1 1 2 2 2 34 81 12 3 90 3 10000 6 0 1 0 1 2 1</pre>	<pre>133</pre>

Exemplo de entrada 2	Exemplo de saída 2
<pre>7 1 1 2 1 2 2 1 7 3 4 1 8 5 10 8 0 2 0 0 1 1 1 1</pre>	<pre>30</pre>