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 (20 e 21 de agosto de 2026).

Olimpíada Brasileira de Informática

OBI2026

Caderno de Tarefas

Modalidade Programação • Nível Júnior • Fase 2

20 e 21 de agosto de 2026

A PROVA TEM DURAÇÃO DE 2 HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

Coordenação:

INSTITUTO DE COMPUTAÇÃO UNICAMP

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 8 páginas (não contando a folha de rosto), numeradas de 1 a 8. 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.

Festas de Aniversário

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

Alice e Bruno são melhores amigos e estão muito animados com seus aniversários que se aproximam. Como cada um faz aniversário em um dia diferente, eles querem descobrir quem vai comemorar primeiro para começar logo os preparativos da festa.

Dados o dia e o mês do aniversário de Alice e o dia e o mês do aniversário de Bruno, calcule quem vai festejar primeiro no ano.

Entrada

A primeira linha da entrada contém um único inteiro D_A, o dia do aniversário de Alice.

A segunda linha da entrada contém um único inteiro M_A, o mês do aniversário de Alice.

A terceira linha da entrada contém um único inteiro D_B, o dia do aniversário de Bruno.

A quarta linha da entrada contém um único inteiro M_B, o mês do aniversário de Bruno.

Saída

A saída deve conter uma única letra: `A` se o aniversário de Alice acontece primeiro, ou `B` se o aniversário de Bruno acontece primeiro. Não coloque nada além disso na saída, senão será considerado errado.

Restrições

- $1 \leq D_A \leq 31$.
- $1 \leq M_A \leq 12$.
- $1 \leq D_B \leq 31$.
- $1 \leq M_B \leq 12$.
- É garantido que as datas são válidas (por exemplo, não haverá 31 de fevereiro).
- É garantido que as festas serão em datas diferentes.

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 (20 pontos):** $M_A = M_B$ (os aniversários são no mesmo mês).
- **Subtarefa 3 (20 pontos):** $D_A = D_B$ (os aniversários são no mesmo dia do mês, porém em meses diferentes).
- **Subtarefa 4 (60 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
7 9 2 11	A

Explicação do exemplo 1: Alice faz aniversário em 7 de setembro e Bruno faz aniversário em 2 de novembro. Nesse caso, a resposta é `A`.

Exemplo de entrada 2	**Exemplo de saída 2**
29 12 15 12	B

Explicação do exemplo 2: Alice faz aniversário em 29 de dezembro e Bruno faz aniversário em 15 de dezembro. Nesse caso, a resposta é `B`. Este exemplo satisfaz as restrições da subtarefa 2.

Exemplo de entrada 3	**Exemplo de saída 3**
10 3 10 8	A

Explicação do exemplo 3: Alice faz aniversário em 10 de março e Bruno faz aniversário em 10 de agosto. Nesse caso, a resposta é `A`. Este exemplo satisfaz as restrições da subtarefa 3.

Passes

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

Durante uma partida de futebol, uma câmera registra, a cada segundo, o número do jogador que está com a bola. Os jogadores do time 1 são identificados por números de 1 a 100, e os jogadores do time 2 são identificados por números de -1 a -100.

Um passe acontece quando, em um segundo, a bola está com um jogador de determinado time e, no segundo seguinte, a bola passa para um **outro** jogador **do mesmo time**. Quando a bola vai de um jogador de um time para um jogador do outro time, isso **não** é considerado passe de nenhum dos dois times.

Por exemplo, se a sequência de posse de bola é $10, 9, 9, -3, -2, -2, -1, -7, 3, 3$, então:

- O time 1 realizou 1 passe: do jogador 10 para o jogador 9.
- O time 2 realizou 3 passes: de -3 para -2, de -2 para -1, e de -1 para -7.

Note que os momentos em que a posse passou de um time para o outro (como de 9 para -3, ou de -7 para 3) não contam como passe de nenhum time.

Dado o registro de N segundos da partida, determine quantos passes realizou cada time.

Entrada

A primeira linha da entrada contém um único inteiro N, a quantidade de segundos registrados.

A segunda linha da entrada contém N inteiros $J_1, J_2, \ldots, J_N$, onde J_i é o número do jogador que está com a bola no segundo i. Números positivos identificam jogadores do time 1, e números negativos identificam jogadores do time 2.

Saída

A saída deve conter exatamente duas linhas. A primeira linha deve conter um inteiro com a quantidade de passes realizados pelo time 1. A segunda linha deve conter um inteiro com a quantidade de passes realizados pelo time 2. Imprima 0 caso um time não tenha realizado nenhum passe.

Restrições

- $3 \leq N \leq 10000$.
- $1 \leq J_i \leq 100$, para todo segundo i em que a bola está com um jogador do time 1.
- $-100 \leq J_i \leq -1$, para todo segundo i em que a bola está com um jogador do time 2.

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 (20 pontos):** $N = 3$.
- **Subtarefa 3 (20 pontos):** Todos os valores de J_i são positivos, ou seja, a bola ficou sempre com jogadores do time 1 durante todo o tempo registrado.
- **Subtarefa 4 (60 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
10 10 9 9 -3 -2 -2 -1 -7 3 3	1 3

Explicação do exemplo 1: Note que este é o exemplo mostrado no enunciado.

Exemplo de entrada 2	**Exemplo de saída 2**
3 5 -2 7	0 0

Explicação do exemplo 2: A bola passou de um time para o outro em todos os momentos, portanto nenhum time realizou passes. Este exemplo satisfaz as restrições da subtarefa 2.

Exemplo de entrada 3	**Exemplo de saída 3**
5 3 3 7 7 1	2 0

Explicação do exemplo 3: O time 1 realizou 2 passes: do jogador 3 para o jogador 7, e do jogador 7 para o jogador 1. Note que, nos momentos em que o mesmo jogador manteve a posse da bola por dois segundos consecutivos (3 e 3; 7 e 7), não houve passe. Este exemplo satisfaz as restrições da subtarefa 3.

Exemplo de entrada 4	**Exemplo de saída 4**
6 1 2 -1 -2 5 -3	1 1

Cinema à Distância

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

As pessoas da N-logônia adoram cinema, porém, há poucas salas de cinema pelo reino, o que dificulta o acesso dessa atividade tão importante para a cultura local. Foi então que Pedro, o rei, decidiu inaugurar uma nova modalidade: Cinema à Distância, onde cada pessoa pode comprar um ingresso e assistir ao filme remotamente.

O sucesso foi instantâneo, mas o sistema de controle não contabiliza quantas pessoas participaram de cada sessão. Por isso, Pedro, o rei, ordenou que você implemente esta funcionalidade.

O sistema registrou que N pessoas compraram ingressos para assistir a um filme. O filme é exibido em M sessões diferentes, e cada uma tem capacidade para C pessoas (mesmo sendo à distância, é necessário ter controle para não vazar o filme). Cada pessoa i comprou seu ingresso no instante T_i (para todo $1 \leq i \leq N$) e cada sessão j tem um horário de início H_j (para todo $1 \leq j \leq M$). Para que uma pessoa consiga entrar em uma sessão, ela precisa ter comprado seu ingresso **antes ou exatamente no** horário de início do filme, ou seja, uma pessoa que comprou o ingresso no instante T_i pode entrar na sessão j somente se $H_j \geq T_i$.

Quando uma pessoa compra um ingresso, ela é alocada na **primeira** sessão disponível em que pode entrar, isto é, a sessão j de menor índice tal que $H_j \geq T_i$ e que ainda não atingiu a capacidade C. Se não existir tal sessão, o ingresso não é vendido, mas fica registrado para as estatísticas do sistema.

Por exemplo, considere $N = 10$ pessoas que compraram ingressos nos instantes $2, 2, 2, 7, 10, 10, 15, 20, 20, 23$, com $M = 3$ sessões com horários de início $3, 9, 25$ e capacidade $C = 4$:

- **Sessão 1** ($H_1 = 3$): as 3 pessoas dos instantes $2, 2, 2$ conseguem entrar, pois $H_1 = 3 \geq 2$. Já a pessoa do instante 7 não consegue entrar nessa sessão, portanto a sessão termina com 3 pessoas.
- **Sessão 2** ($H_2 = 9$): a pessoa do instante 7 entra nessa sessão ($9 \geq 7$). Já as pessoas do instante 10 não conseguem, pois $H_2 = 9 < 10$. Assim a sessão termina com 1 pessoa.
- **Sessão 3** ($H_3 = 25$): as pessoas dos instantes $10, 10, 15, 20$ entram, preenchendo a capacidade máxima (4 pessoas). Perceba que as pessoas dos instantes 20 e 23 não conseguem ingresso, pois a sessão já está cheia e não há outras sessões.

Dado o número N de pessoas, o número M de sessões, a capacidade C de cada sessão, os instantes de compra das pessoas (T_i para todo $1 \leq i \leq N$) e os horários de início de cada sessão (H_j para todo $1 \leq j \leq M$), determine quantas pessoas assistiram ao filme em cada sessão.

Entrada

A primeira linha da entrada contém três inteiros N, M e C, indicando respectivamente o número de pessoas, o número de sessões e a capacidade de cada sessão.

A segunda linha contém N inteiros $T_1, T_2, \ldots, T_N$, os instantes em que cada pessoa comprou seu ingresso, em ordem **não decrescente** ($T_i \leq T_{i+1}$ para todo $1 \leq i < N$).

A terceira linha contém M inteiros $H_1, H_2, \ldots, H_M$, os horários de início de cada sessão, em ordem **não decrescente** ($H_j \leq H_{j+1}$ para todo $1 \leq j < M$).

Saída

A saída deve conter uma única linha com M inteiros separados por espaços. O j-ésimo inteiro deve ser a quantidade de pessoas que assistirão ao filme na sessão j.

Restrições

- $1 \leq N \leq 10^5$.
- $1 \leq M \leq 10^5$.
- $1 \leq C \leq 10^5$.
- $1 \leq T_i \leq 10^9$, para todo $1 \leq i \leq N$.
- $1 \leq H_j \leq 10^9$, para todo $1 \leq j \leq M$.
- $T_i \leq T_{i+1}$, para todo $1 \leq i < N$.
- $H_j \leq H_{j+1}$, para todo $1 \leq j < M$.

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):** $T_i = 1$ para todo $1 \leq i \leq N$.
- **Subtarefa 3 (14 pontos):** $M = 1$.
- **Subtarefa 4 (32 pontos):** $N, M, C \leq 1000$.
- **Subtarefa 5 (37 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**

`10 3 4`
`2 2 2 7 10 10 15 20 20 23`
`3 9 25`
	`3 1 4`

Explicação do exemplo 1: Este é o exemplo apresentado no enunciado. A sessão 1 ($H_1 = 3$) recebe as 3 pessoas que compraram no instante 2 (pois $3 \geq 2$). A sessão 2 ($H_2 = 9$) recebe apenas a pessoa do instante 7 (pois $9 \geq 7$, mas $9 < 10$). A sessão 3 ($H_3 = 25$) recebe as 4 pessoas dos instantes $10, 10, 15, 20$, atingindo a capacidade máxima. As duas últimas pessoas não conseguem ingresso.

Exemplo de entrada 2	**Exemplo de saída 2**

`10 2 6`
`1 1 1 1 1 1 1 1 1 1`
`2 20`
	`6 4`

Explicação do exemplo 2: Como todos compraram no instante 1, qualquer sessão é acessível. A sessão 1 fica cheia com 6 pessoas. As 4 restantes vão para a sessão 2. Este exemplo satisfaz as restrições da Subtarefa 2.

Exemplo de entrada 3	**Exemplo de saída 3**
5 1 10 1 3 5 8 11 6	3

Explicação do exemplo 3: O filme começa no instante $H_1 = 6$. As pessoas dos instantes $1, 3$ e 5 conseguem entrar (pois $6 \geq T_i$). As dos instantes 8 e 11 não conseguem, pois compraram o ingresso depois do início do filme ($T_i > H_1$). A sessão tem capacidade $C = 10$, então as 3 pessoas que chegaram a tempo entram sem problema. Este exemplo satisfaz as restrições da Subtarefa 3.

Exemplo de entrada 4	**Exemplo de saída 4**
4 2 3 7 8 9 10 1 3	0 0

Explicação do exemplo 4: As sessões começam nos instantes 1 e 3, mas as pessoas só compraram ingressos a partir do instante 7. Portanto, nenhuma pessoa consegue um ingresso válido, e a resposta é 0 para cada sessão.