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 Sênior • Fase 2

20 e 21 de agosto de 2026

A PROVA TEM DURAÇÃO DE 3 HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

Coordenação:

INSTITUTO DE COMPUTAÇÃO UNICAMP

v1.0

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.

Adivinha o Número

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

Na aula de matemática, os alunos estavam tão agitados que a professora Lúcia decidiu propor uma brincadeira para acalmá-los: **Adivinha o Número**

As regras são simples:

- A professora escolhe um número secreto P, sem revelar para ninguém.
- Os alunos Ana e Beto, ao mesmo tempo, chutam um número: Ana chuta A e Beto chuta B.
- A professora então revela o número P.
- Ganha quem chegou mais perto do número da professora. Se os dois ficarem à mesma distância, é empate.

Dados os chutes de Ana e Beto e o número escolhido pela professora, descubra quem ganhou a brincadeira.

Entrada

A primeira linha da entrada contém um único inteiro A, o chute de Ana.

A segunda linha da entrada contém um único inteiro B, o chute de Beto.

A terceira linha da entrada contém um único inteiro P, o número escolhido pela professora.

Saída

A saída deve conter uma única letra: `A` se Ana ganhou, `B` se Beto ganhou, ou `E` se houve empate. Não coloque nada além disso na saída, senão será considerado errado.

Restrições

- $1 \leq A \leq 100$.
- $1 \leq B \leq 100$.
- $1 \leq P \leq 100$.

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 (20 pontos):** $A \leq P \leq B$.
- **Subtarefa 3 (20 pontos):** $A \leq B$.
- **Subtarefa 4 (60 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
10 5 8	A

Explicação do exemplo 1: Ana chutou 10, Beto chutou 5 e a professora revelou o número 8. Nesse caso, a resposta é `A`.

Exemplo de entrada 2	**Exemplo de saída 2**
1 9 5	E

Explicação do exemplo 2: Ana chutou 1, Beto chutou 9 e a professora revelou o número 5. Nesse caso, a resposta é `E`. Este exemplo satisfaz as restrições das subtarefas 2 e 3.

Exemplo de entrada 3	**Exemplo de saída 3**
2 3 20	B

Explicação do exemplo 3: Ana chutou 2, Beto chutou 3 e a professora revelou o número 20. Nesse caso, a resposta é `B`. Este exemplo satisfaz as restrições da subtarefa 3.

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.

Separador de Produtos

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

A fábrica onde você trabalha possui um robô separador recém-adquirido, projetado para otimizar o fluxo de produtos que devem ser embalados. No entanto, os engenheiros da fábrica suspeitam que o robô esteja com um mau funcionamento em seu sistema de distribuição, e você foi chamado para verificar se ele está separando os produtos corretamente.

O robô recebe uma esteira com N produtos, cada um com seu respectivo peso, e deve distribuí-los entre F filas de embalagem, sempre em ordem. O manual do robô descreve seu algoritmo de funcionamento da seguinte forma:

Para cada produto (do índice 1 até o N), o robô adiciona o produto atual à fila que possui no momento o **menor peso total**. Caso haja um empate entre duas ou mais filas com o mesmo peso total mínimo, o robô deve escolher a fila de **menor índice** dentre elas.

Dada a lista dos N produtos na ordem em que chegam, você deve imprimir o resultado esperado desse processo de separação nas F filas.

Entrada

A primeira linha da entrada contém dois inteiros, N e F, o número de produtos e o número de filas, respectivamente. A segunda linha da entrada contém N inteiros positivos, os pesos P_i de cada produto, na ordem em que aparecem.

Saída

Seu programa deve imprimir F linhas, correspondendo ao conteúdo de cada fila após a separação de todos os produtos, na ordem da fila 1 até a fila F. Para cada linha, liste os pesos dos produtos daquela fila, na ordem em que foram adicionados, separados por um espaço.

Restrições

- $2 \leq F \leq N \leq 10^5$
- $1 \leq P_i \leq 10^4$ para todo $1 \leq i \leq N$

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):** Todos os pesos são iguais a 1.
- **Subtarefa 3 (19 pontos):** $F = 2$.
- **Subtarefa 4 (33 pontos):** $N \leq 1000$.
- **Subtarefa 5 (30 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
10 4 7 3 1 6 5 2 2 3 2 1	7 1 3 2 2 1 5 3 6 2

Explicação do exemplo 1:

- O produto 1 com peso 7 vai para a fila 1, pois todas estão vazias e ela possui o menor índice.
- O produto 2 com peso 3 vai para a fila 2, que está vazia e tem o menor índice.
- O produto 3 com peso 1 vai para a fila 3.
- O produto 4 com peso 6 vai para a fila 4.
- Neste momento os pesos das filas são: 7, 3, 1, 6. O produto 5 com peso 5 vai para a fila 3 (menor peso, 1). O peso da fila 3 passa a ser 6.
- Neste momento os pesos das filas são: 7, 3, 6, 6. O produto 6 com peso 2 vai para a fila 2. Peso da fila 2 passa a 5.
- Neste momento os pesos das filas são: 7, 5, 6, 6. O produto 7 com peso 2 vai para a fila 2. Peso da fila 2 passa a 7.
- Neste momento os pesos das filas são: 7, 7, 6, 6. O produto 8 com peso 3 vai para a fila 3 (empate entre as filas 3 e 4, logo a de menor índice é a 3). Peso da fila 3 passa a ser 9.
- Neste momento os pesos das filas são: 7, 7, 9, 6. O produto 9 com peso 2 vai para a fila 4. Peso da fila 4 passa a ser 8.
- Neste momento os pesos das filas são: 7, 7, 9, 8. O produto 10 com peso 1 vai para a fila 1 (empate entre as filas 1 e 2, logo a de menor índice é a 1). Peso da fila 1 passa a ser 8.

Exemplo de entrada 2	**Exemplo de saída 2**
6 3 1 1 1 1 1 1	1 1 1 1 1 1

Explicação do exemplo 2: Neste exemplo temos 6 produtos, todos de peso 1. Eles são distribuídos igualmente pelas 3 filas disponíveis, respeitando sempre o menor peso e, em caso de empate, o menor índice. Este exemplo satisfaz as restrições da subtarefa 2.

Exemplo de entrada 3	**Exemplo de saída 3**
5 2 10 3 2 5 6	10 6 3 2 5

Explicação do exemplo 3: Este exemplo satisfaz as restrições da subtarefa 3.

Loja Virtual

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

Beto é um analista de dados na Nlogônia Store, uma gigantesca loja virtual. O sistema da loja exibe o catálogo de produtos para os clientes em um formato de grade paginada. Existem N páginas no total, e cada página exibe exatamente M produtos, listados horizontalmente da esquerda para a direita.

Uma característica nativa do site é que o catálogo está ordenado estritamente pela data de lançamento do produto. O tempo avança da esquerda para a direita dentro de uma mesma página e, naturalmente, flui de uma página para a seguinte. Ou seja, o produto na primeira posição da página $T+1$ foi lançado cronologicamente logo após o último produto da página T (para $1 \leq T < N$).

Enquanto analisava o banco de dados, Beto percebeu uma coincidência curiosa: ao selecionar áreas retangulares específicas na grade do catálogo, os preços dos produtos dentro dessa seleção, quando lidos na ordem natural do tempo (da esquerda para a direita, página por página), formavam uma sequência estritamente crescente.

Beto quer estudar esse padrão a fundo, mas o catálogo é enorme. Por isso, ele pediu a sua ajuda. Sua tarefa é escrever um programa que, dada a matriz representando os preços dos produtos no catálogo, encontre o tamanho da maior seleção possível (ou seja, a área da maior submatriz) que atenda a essa coincidência.

A submatriz é formada escolhendo-se uma página inicial e uma página final, bem como uma coluna inicial e uma coluna final. Ela é válida se, ao linearizar os preços selecionados seguindo o fluxo natural das páginas, a sequência gerada for estritamente crescente.

Entrada

A primeira linha contém dois inteiros N e M, representando o número de páginas e o número de produtos exibidos por página, respectivamente.

As próximas N linhas contêm M inteiros positivos cada. Onde o j-ésimo inteiro da i-ésima linha, $P_{i,j}$, representa o preço do produto localizado na coluna j da página i.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, a quantidade máxima de produtos em uma submatriz válida.

Restrições

- $1 \leq N \leq 800$.
- $1 \leq M \leq 800$.
- $1 \leq P_{i,j} \leq 10^7$, para todo $1 \leq i \leq N$ e $1 \leq j \leq M$.

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 (12 pontos):** $N = 1$.
- **Subtarefa 3 (16 pontos):** $N, M \leq 20$.
- **Subtarefa 4 (19 pontos):** $N, M \leq 100$.
- **Subtarefa 5 (22 pontos):** $N, M \leq 500$.
- **Subtarefa 6 (31 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
3 4 5 1 2 6 8 3 5 9 2 5 8 10	4

Explicação do exemplo 1: Neste exemplo, a resposta é 4. Há exatamente duas seleções válidas com 4 produtos:

- A seleção formada pelas linhas 1 a 2 e colunas 2 a 3 (usando índices a partir de 1), ao linearizar, obtemos a sequência $1, 2, 3, 5$, que é estritamente crescente.
- A seleção formada pela linha 3 e colunas 1 a 4: ao linearizar, obtemos a sequência $2, 5, 8, 10$, que é estritamente crescente.

Note que não é possível selecionar as linhas 1 a 3 e colunas 2 a 3, pois a sequência combinada seria $1, 2, 3, 5, 5, 8, 10$, que não é estritamente crescente (o valor 5 se repete).

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

Explicação do exemplo 2: Como $N = 1$, a grade tem apenas uma linha. Neste exemplo a maior quantidade de produtos numa submatriz válida é 3, correspondendo ao intervalo de colunas de 2 a 4 da única linha, que ao ser linearizada resulta em $1, 4, 5$.

Exemplo de entrada 3	**Exemplo de saída 3**
6 8 500 100 200 300 400 500 600 700 800 900 100 200 300 400 500 600 999 1 2 3 4 5 6 999 998 7 8 9 10 11 12 997 700 600 500 400 300 200 100 800 150 160 170 180 190 200 210 220	12