Olimpíada Brasileira de Informática

OBI2025

Caderno de Tarefas

Modalidade Programação • Nível 2 • Fase 1

12 a 14 de Junho de 2025

A PROVA TEM DURAÇÃO DE 2 horas

Promoção:

Sociedade Brasileira de Computação

Apoio:

Coordenação:

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.

Festa Junina

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

A escola de Luísa está se preparando para a festa junina deste ano. Para este grande evento, a cozinha da escola precisa de ingredientes para preparar pratos típicos como canjica e pamonha. Além disso, os alunos participarão de uma dança de quadrilha, para a qual eles precisam de roupas tradicionais como camisas xadrez e chapéus de palha.

A professora de Luísa pediu ajuda a ela para comprar ingredientes e roupas para a festa. Luísa irá comprar as roupas na lojinha do bairro e os ingredientes no supermercado. A escola, o supermercado e a lojinha estão localizados na mesma rua reta. A posição de cada um destes três prédios pode ser representada por um inteiro indicando a distância (em metros) do prédio ao início da rua. A distância percorrida para ir de um prédio a outro é dada pela diferença entre as posições deles.

Luísa atualmente está na escola e precisa visitar o supermercado e a lojinha, em qualquer ordem. Em seguida, ela deve voltar com as compras para a escola. Ajude Luísa a descobrir qual a distância que ela precisa percorrer, no mínimo, para fazer todas as compras e voltar para a escola.

Por exemplo, se as posições da escola, supermercado e lojinha são 10, 5 e 13, respectivamente, Luísa pode ir primeiro ao supermercado, percorrendo $10 - 5 = 5$ metros, então ir do supermercado à lojinha, percorrendo $13 - 5 = 8$ metros, e então voltar para a escola, percorrendo $13 - 10 = 3$ metros. No total, a distância que ela vai percorrer será $5 + 8 + 3 = 16$ metros. É possível verificar que, caso ela escolha visitar primeiro a lojinha e depois o supermercado, ela também percorrerá 16 metros no total.

Entrada

A entrada possui três linhas, cada uma contendo um único inteiro:

- a primeira linha contém o inteiro E, a posição da escola;
- a segunda linha contém o inteiro S, a posição do supermercado;
- a terceira linha contém o inteiro L, a posição da lojinha.

Saída

Seu programa deverá imprimir uma única linha contendo um único inteiro, a distância total em metros que Luísa precisa percorrer.

Restrições

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

- $0 \leq E,\ S,\ L \leq 1\,000$
- As posições dos três locais são distintas.

Informações sobre a pontuação

A tarefa vale 100 pontos.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
10 5 13	16

Exemplo de entrada 2	**Exemplo de saída 2**
4 25 17	42

Exemplo de entrada 3	**Exemplo de saída 3**
1000 0 500	2000

Dieta

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

O gato Garfield comeu lasanhas demais nos últimos dias, o que está afetando seu metabolismo. Por isso, seu dono John decidiu colocá-lo em uma dieta muito rígida.

Seguindo as instruções do método SBC (*Seleção Benéfica de Calorias*), John definiu um limite M de calorias que o gato poderia consumir diariamente. Para não perder as contas de quantas calorias Garfield já consumiu no dia, John observa o rótulo das lasanhas e anota em uma lista as quantidades em gramas de proteínas, gorduras e carboidratos presentes em cada uma das N refeições do gato.

Para calcular quantas calorias Garfield já consumiu, John utiliza a seguinte conversão:

- 1 grama de proteína tem 4 calorias.
- 1 grama de gordura tem 9 calorias.
- 1 grama de carboidrato tem 4 calorias.

John é um humano e consegue calcular isso facilmente. Porém, Garfield é apenas um gato que gosta de comer. Portanto, dada a lista de refeições que Garfield já fez, ajude o gato a saber qual o máximo de calorias que ele ainda pode consumir, sem exceder o limite M determinado.

Entrada

A primeira linha da entrada contém dois inteiros N e M: a quantidade de refeições na lista de John e o limite de calorias, respectivamente.

Cada uma das N linhas seguintes contém três inteiros, P, G e C: as quantidades (em gramas) de proteínas, gorduras e carboidratos, respectivamente, de uma refeição na lista de John.

Saída

Seu programa deverá imprimir uma única linha contendo um único inteiro: a quantidade máxima de calorias que Garfield ainda pode consumir sem exceder o limite M.

Restrições

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

- $1 \leq N \leq 30$
- $1 \leq M \leq 300\,000$
- $0 \leq P,\ G,\ C \leq 500$
- O total de calorias nas refeições na lista de John não excede o limite 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 (30 pontos)**: $N = 1$.
- **Subtarefa 3 (70 pontos)**: Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
3 2000 65 15 20 40 20 25 50 10 35	655

Exemplo de entrada 2	**Exemplo de saída 2**
1 3700 50 300 200	0

Cafeteria

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

Felipe trabalha em uma cafeteria especializada em café espresso com leite. O chefe dele criou uma promoção na qual os clientes recebem um desconto caso tragam suas próprias xícaras, evitando o uso de materiais descartáveis. A promoção se tornou muito popular, o que é ótimo para o meio ambiente mas dificultou o trabalho de Felipe, pois cada cliente possui uma xícara de um tamanho diferente. Além disso, cada cliente prefere uma quantidade diferente de leite na bebida.

Ao fazer um pedido, o cliente indica para Felipe dois números: o volume mínimo A (em mililitros) e o volume máximo B (em mililitros) de leite que ele deseja em sua bebida. O cliente indica também a capacidade C (também em mililitros) de sua xícara.

Para preparar o pedido, Felipe primeiro insere a xícara na máquina de café espresso e indica o número de doses de espresso que deseja. Cada dose possui D mililitros de café. Felipe pode preparar quantas doses desejar, porém, por segurança, a máquina não permite que ele remova a xícara enquanto uma dose está sendo preparada. Deste modo, ele só consegue preparar volumes de café que são múltiplos de D. Por exemplo, se $D = 40$, os volumes que ele consegue consegue preparar são 40, 80, 120, etc.

Depois de remover a xícara da máquina, Felipe adiciona leite de modo a enchê-la completamente, ou seja, o volume total de café com leite deve ser exatamente C. Ele gostaria de escolher o número de doses de espresso e o volume de leite de modo a satisfazer as preferências do cliente. Porém, dependendo dos parâmetros A, B, C e D, isto pode ser possível ou não.

Por exemplo, suponha que cada dose de espresso possua $D = 30$ ml. Considere dois clientes:

- O cliente 1 possui uma xícara com capacidade $C = 200$ ml e deseja entre $A = 130$ ml e $B = 150$ ml de leite. Para este cliente, Felipe pode preparar duas doses de espresso, resultando em $2 \times 30 = 60$ ml de café, e completar a xícara com $200 - 60 = 140$ ml de leite.
- O cliente 2 possui uma xícara com capacidade $C = 250$ ml e deseja no mínimo $A = 200$ ml e no máximo $B = 210$ ml de leite. Felipe não consegue satisfazer as preferências deste cliente: se ele preparar apenas uma dose de espresso, precisará completar a xícara com $250 - 30 = 220$ ml de leite; por outro lado, se ele preparar duas ou mais doses, sobra espaço para no máximo $250 - 2 \times 30 = 190$ ml de leite. Em nenhum caso, ele adiciona entre 200 e 210 ml de leite.

Escreva um programa para ajudar Felipe: dados os volumes A, B e C especificados por um cliente e o volume D de cada dose de espresso feita pela máquina, determine se Felipe consegue escolher o número de doses de espresso tal que o volume de leite na xícara atenda às preferências do cliente.

Entrada

A entrada possui quatro linhas, cada uma contendo um único inteiro:

- a primeira linha contém o volume mínimo A de leite (em ml) que o cliente deseja;
- a segunda linha contém o volume máximo B de leite (em ml) que o cliente deseja;
- a terceira linha contém a capacidade C (em ml) da xícara;
- a quarta linha contém o volume D (em ml) de café em cada dose preparada pela máquina.

Saída

Seu programa deverá imprimir uma única linha contendo um único caractere: caso Felipe consiga satisfazer as preferências do cliente, imprima o caractere `S` (a letra S maiúscula). Caso contrário, imprima o caractere `N` (a letra N maiúscula).

Restrições

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

- $100 \leq C \leq 500$
- $0 \leq A \leq B < C$
- $10 \leq D \leq 100$

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 (25 pontos):** $A = B$.
- **Subtarefa 3 (30 pontos):** $A = 0$.
- **Subtarefa 4 (45 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
130 150 200 30	S

Exemplo de entrada 2	**Exemplo de saída 2**
200 210 250 30	N

Exemplo de entrada 3	**Exemplo de saída 3**
120 120 295 35	S

Exemplo de entrada 4	**Exemplo de saída 4**
0 10 500 90	N

Gráfico de Barras

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

Caique é o mais novo estagiário da OBI (*Organização de Brinquedos Infantis*) e está encarregado de preparar os slides que serão apresentados na próxima reunião de vendas do departamento. A fim de avaliar os futuros investimentos da empresa, o chefe do garoto pediu que ele fizesse um gráfico sobre a popularidade de diversos brinquedos.

Para conseguir esses dados, a OBI realizou previamente uma pesquisa com várias crianças, na qual cada uma escolheu seu brinquedo favorito dentre N opções disponíveis. Os resultados da pesquisa foram organizados em uma lista de N valores, onde o i-ésimo desses valores, X_i, indica quantos participantes da pesquisa preferem o i-ésimo brinquedo. Caique já possui essa lista e deve criar um gráfico de barras com base nessas informações.

Ele deve seguir a seguinte formatação padrão da empresa:

- O gráfico deve conter N colunas de mesma altura H compostas de números 0 e 1, representando os valores na ordem da lista.
- Não deve existir nenhuma linha completamente preenchida por números 0, ou seja, H deve ser igual ao maior valor X_i da lista.
- A base da i-ésima coluna deve ser composta de X_i números iguais a 1, enquanto os outros $H - X_i$ números do topo devem ser iguais a 0.

Por exemplo, suponha que $N = 4$ e a lista seja $4,\ 2,\ 5,\ 3$. Nesse caso, $H = 5$ e o gráfico de barras deve ser:

0	0	1	0
1	0	1	0
1	0	1	1
1	1	1	1
1	1	1	1

Caique ainda está muito ocupado com outros pedidos de seu chefe e pediu sua ajuda. Portanto, dada a lista de popularidade dos N brinquedos, ajude-o a construir um gráfico que siga as especificações da OBI.

Entrada

A primeira linha da entrada contém um único inteiro N, a quantidade de brinquedos da pesquisa realizada pela OBI. A segunda linha da entrada contém N inteiros, o i-ésimo desses valores, X_i, indica a quantidade de participantes da pesquisa que preferem o i-ésimo brinquedo.

Saída

A saída deve conter o gráfico de barras conforme especificado pela OBI, portanto, deve ser composta por H linhas, onde cada uma delas deve ser composta por N números, cada um sendo 0 ou 1, formando o gráfico desejado pela OBI.

Restrições

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

- $1 \leq N \leq 100$
- $0 \leq X_i \leq 100$, para todo $1 \leq i \leq N$
- Existe algum elemento não-nulo na lista, ou seja, $\max(X_i) > 0$

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):** $X_i \leq 2$, para todo $1 \leq i \leq N$.
- **Subtarefa 3 (70 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
4 4 2 5 3	0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1

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

Exemplo de entrada 3	**Exemplo de saída 3**
8 1 2 3 4 5 6 7 8	0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1