

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 (3 de outubro de 2026).



Olimpíada Brasileira de Informática

OBI2026

Caderno de Tarefas

Modalidade Programação • Nível Mirim • Fase 3

3 de outubro de 2026

A PROVA TEM DURAÇÃO DE 4 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 12 páginas (não contando a folha de rosto), numeradas de 1 a 12. 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 Pascal devem ser arquivos com sufixo *.pas*; 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).
- Para tarefas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada tarefa.
- Ao final da prova, para cada solução que você queira submeter para correção, copie o arquivo fonte para o seu diretório de trabalho ou pen-drive, conforme especificado pelo seu professor.
- 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 Pascal: *readln*, *read*, *writeln*, *write*;
 - 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.

Cupom de Desconto

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

Maria é uma criança muito esperta que adora aproveitar promoções na sua loja de departamentos favorita. Hoje, a loja lançou uma promoção especial chamada “Par Perfeito”.

A regra da promoção é simples: o cliente deve escolher exatamente dois produtos. Se a diferença de preço (maior menos o menor) entre esses dois produtos for menor ou igual a K , o cliente ganha um cupom de desconto no valor fixo de R\$ 10,00 (para ser usado em compras futuras). Se a diferença for maior que K , o cliente não ganha nenhum cupom.

Maria separou exatamente 4 produtos que deseja muito comprar, e precisa separá-los de forma a maximizar a quantidade de cupons de desconto que ela irá receber.

Ela pediu sua ajuda, dados os preços dos 4 produtos e o valor K da promoção, determine qual é o maior número de cupons de desconto que ela pode obter.

Entrada

A entrada é composta por cinco linhas. As quatro primeiras linhas contêm os preços dos produtos que Maria quer comprar, representados por V_1, V_2, V_3 e V_4 . A quinta linha contém um número K , que representa a diferença máxima permitida para ganhar o cupom.

Saída

A saída deve conter uma única linha com um inteiro, representando a quantidade máxima de cupons que Maria pode conseguir.

Restrições

- $10 \leq V_1, V_2, V_3, V_4, K \leq 1000$.

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):** $K = 1000$.
- **Subtarefa 3 (20 pontos):** $V_2 = V_3 = V_4$ (os três últimos valores são iguais).
- **Subtarefa 4 (60 pontos):** Sem restrições adicionais.

Dica de pontuação: se o seu programa funcionar corretamente apenas para as restrições de uma subtarefa, você receberá os pontos daquela subtarefa. Pode valer a pena entregar uma solução parcial!

Exemplos

Exemplo de entrada 1	Exemplo de saída 1
25 15 10 35 10	2

Explicação do exemplo 1: Para conseguir 2 cupons, Maria pode agrupar os produtos de preço 10 e 15 ($15 - 10 = 5 \leq K = 10$), ganhando um cupom, e também os produtos de preço 25 e 35 ($35 - 25 = 10 \leq K = 10$), ganhando o segundo cupom.

Exemplo de entrada 2	Exemplo de saída 2
300 100 400 500 10	0

Explicação do exemplo 2: Não é possível agrupar os produtos de nenhuma forma para conseguir cupons.

Exemplo de entrada 3	Exemplo de saída 3
300 100 400 500 1000	2

Explicação do exemplo 3: Para conseguir 2 cupons, Maria pode agrupar os produtos de preço 300 e 500 ($500 - 300 = 200 \leq K = 1000$), ganhando um cupom, e também os produtos de preço 100 e 400 ($400 - 100 = 300 \leq K = 1000$), ganhando o segundo cupom. Este exemplo satisfaz as restrições da sub tarefa 2.

Exemplo de entrada 4	Exemplo de saída 4
111 100 100 100 10	1

Explicação do exemplo 4: Para conseguir um cupom, Maria pode agrupar dois produtos de preço 100 ($100 - 100 = 0 \leq K = 10$), percebe que, com o que sobra ela não consegue ganhar outro cupom, pois ao agrupar 100 com 111 a diferença é maior que K . Este exemplo satisfaz as restrições da sub tarefa 3.

Cozinheiro Real

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

Como todos sabem o casamento do Rei de Nlogônia está próximo e para agradar seu povo, além de ter muito arroz na festa (pois é símbolo de prosperidade), haverá cachorro-quente para todo mundo!

Pedro, o Rei, sabe que há uma discussão fervorosa se deve ou não colocar purê de batata no cachorro-quente, ele já ouviu falar até de um reino onde a população era dividida em dois grupos antagônicos: os Puristas, que defendem o cachorro-quente original, sem purê de batata e a dos Purê-istas, que adoram cachorro-quente com purê de batata. Para evitar brigas em seu casamento, ele solicitou a preferência de cada morador do reino.

O cozinheiro está desesperado pois ele precisa ir ao mercado comprar todos os ingredientes, e ele não sabe quanto de batata deve comprar, além disso ele tem medo de entregar um cachorro-quente errado para alguma pessoa. Dadas as preferências dos habitantes de Nlogônia, ajude o cozinheiro calculando a quantidade dos que preferem o cachorro-quente sem purê, a quantidade dos que preferem o cachorro-quente com purê, e as listas dos habitantes de cada uma dessas preferências.

Entrada

A entrada é composta por duas linhas:

- A primeira linha contém um inteiro N , representando o número de habitantes de Nlogônia. As pessoas são numeradas de 1 a N .
- A segunda linha contém N inteiros $P_1, P_2, \dots, P_N$, onde o valor P_i indica a preferência da pessoa i . Cada valor P_i pode ser 0 (prefere sem purê) ou 1 (prefere com purê).

Saída

A saída deve conter exatamente quatro linhas, no seguinte formato:

- A primeira linha deve conter a quantidade de pessoas que preferem cachorro-quente sem purê (identificadas pelo valor 0).
- A segunda linha deve conter a lista dos números das pessoas que preferem cachorro-quente sem purê.
- A terceira linha deve conter a quantidade de pessoas que preferem cachorro-quente com purê (identificadas pelo valor 1).
- A quarta linha deve conter a lista dos números das pessoas que preferem cachorro-quente com purê.

Restrições

- $2 \leq N \leq 10^5$.
- $0 \leq P_i \leq 1$ para todo $1 \leq i \leq N$.
- É garantido que haverá pelo menos uma pessoa que prefere cachorro-quente sem purê, e pelo menos uma pessoa que prefere cachorro-quente com purê.

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 (15 pontos):** $N = 2$.
- **Subtarefa 3 (29 pontos):** $N \leq 100$.
- **Subtarefa 4 (25 pontos):** Há apenas uma pessoa que prefere cachorro-quente com purê.
- **Subtarefa 5 (31 pontos):** Sem restrições adicionais.

Dica de pontuação: se o seu programa funcionar corretamente apenas para as restrições de uma subtarefa, você receberá os pontos daquela subtarefa. Pode valer a pena entregar uma solução parcial!

Exemplos

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

Explicação do exemplo 1: Neste exemplo, temos $N = 7$ pessoas, com as seguintes preferências:

- A pessoa 1 prefere cachorro-quente sem purê, pois $P_1 = 0$.
- A pessoa 2 prefere cachorro-quente com purê, pois $P_2 = 1$.
- A pessoa 3 prefere cachorro-quente com purê, pois $P_3 = 1$.
- A pessoa 4 prefere cachorro-quente sem purê, pois $P_4 = 0$.
- A pessoa 5 prefere cachorro-quente sem purê, pois $P_5 = 0$.
- A pessoa 6 prefere cachorro-quente com purê, pois $P_6 = 1$.
- A pessoa 7 prefere cachorro-quente sem purê, pois $P_7 = 0$.

Portanto temos 4 pessoas que preferem sem purê: 1, 4, 5 e 7; e 3 pessoas que preferem com purê: 2, 3 e 6.

Exemplo de entrada 2	Exemplo de saída 2
2 0 1	1 1 1 2

Explicação do exemplo 2: Este exemplo satisfaz as restrições da subtarefa 2.

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

Explicação do exemplo 3: Este exemplo satisfaz as restrições da subtarefa 4.

Rebobinar

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

Elisa é uma influencer muito famosa que acabou de publicar o último episódio de uma série de enorme sucesso em seu canal. No total, a série publicada é composta por N pequenas cenas. No arquivo de gravação original (antes de qualquer edição), essas cenas estavam perfeitamente organizadas em ordem cronológica: a posição 1 continha a cena 1, a posição 2 continha a cena 2, e assim por diante, até a posição N com a cena N .

Durante o processo de edição da série, para criar efeitos visuais, *flashbacks* e momentos de suspense, Elisa usou a ferramenta “Rebobinar” Q vezes. A cada utilização da ferramenta, ela selecionava um trecho contínuo da série, da posição L até a posição R , e fazia o programa de edição inverter totalmente a ordem daquele bloco (a cena que estava na posição L trocava de lugar com a da posição R , a da posição $L + 1$ trocava com a de $R - 1$, e assim por diante).

Pouco tempo após o lançamento, Elisa recebeu uma notificação de direitos autorais avisando que havia uma imagem não autorizada exatamente na posição K da série completa publicada, isto é, após as edições.

Elisa possui todo o material de gravação original salvo em seu computador, mas precisa da sua ajuda para identificar a cena correta que gerou o problema. Como a ordem mudou bastante durante as edições, sua tarefa é descobrir qual era o número da cena que acabou indo parar na posição K da série publicada.

Entrada

A primeira linha da entrada contém três inteiros N , K e Q , em que N é o número de cenas, K é a posição que gerou o problema e Q é o número de vezes que a ferramenta Rebobinar foi usada.

Cada uma das Q linhas seguintes contém dois inteiros L_i e R_i , indicando que a i -ésima utilização da ferramenta inverteu o trecho da posição L_i até a posição R_i .

Saída

A saída deve conter um único inteiro: o número da cena original que está na posição K da série publicada.

Restrições

- $2 \leq N \leq 10^9$.
- $1 \leq K \leq N$.
- $1 \leq Q \leq 3 \times 10^5$.
- $1 \leq L_i < R_i \leq N$, para todo $1 \leq i \leq Q$.

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 (16 pontos):** $N \leq 10^3$, $Q \leq 10^3$ e $R_i = L_i + 1$ para todo $1 \leq i \leq Q$ (cada operação troca apenas dois elementos adjacentes).
- **Subtarefa 3 (28 pontos):** $N \leq 10^3$ e $Q \leq 10^3$.
- **Subtarefa 4 (19 pontos):** $Q = 1$.
- **Subtarefa 5 (37 pontos):** Sem restrições adicionais.

Dica de pontuação: se o seu programa funcionar corretamente apenas para as restrições de uma subtarefa, você receberá os pontos daquela subtarefa. Pode valer a pena entregar uma solução parcial!

Exemplos

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

Explicação do exemplo 1: Começamos com as cenas na ordem 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

Após inverter o trecho [4, 8]: 1, 2, 3, **8, 7, 6, 5**, 4, 9, 10.

Após inverter o trecho [3, 6]: 1, 2, **6, 7, 8, 3**, 5, 4, 9, 10.

Após inverter o trecho [4, 9]: 1, 2, 6, **9, 4, 5, 3, 8, 7**, 10.

Como $K = 6$, a resposta é o elemento na posição 6, que é 5.

Exemplo de entrada 2	Exemplo de saída 2
8 3 4 2 3 3 4 2 3 3 4	2

Explicação do exemplo 2: Este exemplo satisfaz as restrições da subtarefa 2.

Começamos com as cenas na ordem 1, 2, 3, 4, 5, 6, 7, 8.

Após inverter o trecho [2, 3]: 1, **3, 2**, 4, 5, 6, 7, 8.

Após inverter o trecho [3, 4]: 1, 3, **4, 2**, 5, 6, 7, 8.

Após inverter o trecho [2, 3]: 1, **4, 3**, 2, 5, 6, 7, 8.

Após inverter o trecho [3, 4]: 1, 4, **2, 3**, 5, 6, 7, 8.

Como $K = 3$, a resposta é o elemento na posição 3, que é 2.

Exemplo de entrada 3	Exemplo de saída 3
15 7 6 2 12 5 10 3 9 7 14 1 8 6 15	4

O Desafio de Neumann

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

Durante um passeio no parque de Nlogônia, o brilhante Neumann viu duas crianças brincando com um jogo chamado “Ligue 4”, onde fichas de diversas cores eram soltas e caíam em uma grade vertical. Neumann não entendeu muito bem o objetivo do jogo em si, mas ficou completamente fascinado pela mecânica da gravidade combinada com a paleta de cores: as bolinhas caíam em linha reta e paravam apenas quando batiam no fundo ou em outra bolinha já posicionada.

Com a mente fervilhando, ele correu até a casa de seu amigo Nash e propôs o seguinte desafio baseado nessa mecânica.

Imagine uma grade gigante com N linhas e M colunas. As linhas são numeradas de 1 a N de cima para baixo, ou seja, a linha 1 é o topo e a linha N é o chão. As colunas são numeradas de 1 a M da esquerda para a direita. Além disso, as bolinhas utilizadas no jogo possuem cores que variam de 1 a K . Serão dados Q comandos, que podem ser de três tipos:

- **Tipo c (Queda):** $c\ j\ k$ - Soltar uma bolinha da cor k na coluna j . Você tem que dizer em qual linha ela parou. Mas se a coluna já estiver completamente cheia até o topo, a bolinha transborda, cai no chão e você deve informar -1.
- **Tipo l (Radar da Linha):** $l\ i$ - Ligar um radar na linha i . Você tem que olhar para essa linha da esquerda para a direita e dizer qual é a primeira coluna que possui alguma bolinha, informando também a sua cor. Se a linha estiver totalmente vazia, você deve informar -1.
- **Tipo p (Radar de Ponto):** $p\ i\ j$ - Ligar um radar direcionado. Você tem que verificar a posição exata na linha i e coluna j . Se houver alguma bolinha lá, você deve informar a cor dela. Se a posição estiver vazia, você deve informar -1.

Nash aceitou o desafio na hora, mas a grade pode ser muito grande. Escreva um programa para ajudar Nash a responder todas as perguntas de Neumann de forma correta e eficiente.

Entrada

A primeira linha contém quatro inteiros N , M , Q e K , representando a quantidade de linhas, colunas, eventos e o número máximo de cores. As Q linhas seguintes descrevem os eventos em um dos seguintes formatos:

- $c\ j\ k$: uma bolinha da cor k foi solta na coluna j .
- $l\ i$: uma pergunta sobre a linha i .
- $p\ i\ j$: uma pergunta sobre a posição (i, j) .

Saída

Para cada evento, imprima a resposta correspondente em uma linha separada. Para os comandos c e p , imprima um único número, conforme descrito anteriormente. Para o comando l , imprima dois números separados por espaço (a coluna e a cor), ou apenas -1 se a linha estiver vazia.

Restrições

- $1 \leq N, M, Q, K \leq 10^5$.

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):** $N = 1, M \leq 100, Q \leq 100$.
- **Subtarefa 3 (13 pontos):** $N = 2, M \leq 100, Q \leq 100$.
- **Subtarefa 4 (21 pontos):** $N, M, Q \leq 100$.
- **Subtarefa 5 (11 pontos):** Há apenas eventos do tipo c.
- **Subtarefa 6 (14 pontos):** $K = 1$, ou seja, todas as bolinhas possuem a cor 1.
- **Subtarefa 7 (24 pontos):** Sem restrições adicionais.

Dica de pontuação: se o seu programa funcionar corretamente apenas para as restrições de uma subtarefa, você receberá os pontos daquela subtarefa. Pode valer a pena entregar uma solução parcial!

Nota: vale ressaltar que não é possível declarar uma matriz de 10^5 linhas por 10^5 colunas pois ultrapassa o limite de memória. Portanto, para as subtarefas onde $N \leq 100$ e $M \leq 100$ é possível declarar a matriz, nas outras, é preciso encontrar algum algoritmo que não declare a matriz.

Exemplos

Exemplo de entrada 1	Exemplo de saída 1
3 4 9 5	3
c 2 3	2
c 2 5	3
c 3 1	2 3
l 3	5
p 2 2	1
c 2 4	-1
c 2 2	2 4
l 1	-1
p 3 4	

Explicação do exemplo 1: Neste exemplo temos $N = 3$ linhas, $M = 4$ colunas, $Q = 9$ comandos, $K = 5$ cores e o funcionamento de cada comando é o seguinte:

1. comando: c 2 3 (soltar uma bolinha de cor 3 na coluna 2)

A bolinha desce até o fundo e para na linha 3.

Saída: 3

	1	2	3	4
1				
2				
3		3		

2. comando: c 2 5 (soltar uma bolinha de cor 5 na coluna 2)

A linha 3 da coluna 2 está ocupada. A bolinha para na linha 2.

Saída: 2

	1	2	3	4
1				
2		5		
3		3		

3. comando: c 3 1 (soltar uma bolinha de cor 1 na coluna 3)

A bolinha desce até o fundo e para na linha 3.

Saída: 3

	1	2	3	4
1				
2		5		
3		3	1	

4. comando: l 3 (pergunta sobre a linha 3)

Olhando a linha 3 da esquerda para a direita, a primeira bolinha encontrada está na coluna 2 e tem a cor 3.

Saída: 2 3

5. comando: p 2 2 (pergunta sobre a posição linha 2 coluna 2)

Existe uma bolinha nesta posição com a cor 5.

Saída: 5

6. comando: c 2 4 (soltar uma bolinha de cor 4 na coluna 2)

As linhas 3 e 2 estão ocupadas. A bolinha para na linha 1.

Saída: 1

	1	2	3	4
1		4		
2		5		
3		3	1	

7. comando: c 2 2 (soltar uma bolinha de cor 2 na coluna 2)

A coluna 2 já possui 3 bolinhas e alcançou o topo. A nova bolinha transborda.

Saída: -1

8. comando: l 1 (pergunta sobre a linha 1)

Olhando a linha 1 da esquerda para a direita, a primeira e única bolinha está na coluna 2, cor 4.

Saída: 2 4

9. comando: p 3 4 (pergunta sobre a posição linha 3 coluna 4)

A posição (3, 4) está totalmente vazia.

Saída: -1

Exemplo de entrada 2	Exemplo de saída 2
1 3 4 5 c 2 4 l 1 c 2 1 p 1 2	1 2 4 -1 4

Explicação do exemplo 2: Este exemplo satisfaz as restrições da subtarefa 2.

Neste exemplo temos $N = 1$ linha, $M = 3$ colunas, $Q = 4$ comandos, $K = 5$ cores e o funcionamento de cada comando é o seguinte:

1. comando: c 2 4 (soltar uma bolinha de cor 4 na coluna 2)

A bolinha desce até o fundo e para na linha 1.

Saída: 1

	1	2	3
1		4	

2. comando: l 1 (pergunta sobre a linha 1)

A primeira bolinha encontrada está na coluna 2 e tem a cor 4.

Saída: 2 4

3. comando: c 2 1 (soltar uma bolinha de cor 1 na coluna 2)

A coluna 2 só possui 1 linha, que já está ocupada. A bolinha transborda.

Saída: -1

4. comando: p 1 2 (pergunta sobre a posição linha 1 coluna 2)

Existe uma bolinha nesta posição com a cor 4.

Saída: 4

Exemplo de entrada 3	Exemplo de saída 3
4 3 5 5 c 1 2 c 2 3 c 1 4 c 3 1 c 1 5	4 4 3 4 2

Explicação do exemplo 3: Este exemplo satisfaz as restrições da subtarefa 5.

Exemplo de entrada 4	Exemplo de saída 4
2 2 3 1 c 2 1 l 2 p 2 2	2 2 1 1

Explicação do exemplo 4: Este exemplo satisfaz as restrições da subtarefa 6.