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 (27 de setembro de 2025).

Olimpíada Brasileira de Informática

OBI2025

Caderno de Tarefas

Modalidade Programação • Nível Júnior • Fase 3

27 de setembro de 2025

A PROVA TEM DURAÇÃO DE 4 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 11 páginas (não contando a folha de rosto), numeradas de 1 a 11. 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.

Sacolas

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

Marcelo, um entusiasta do churrasco, decidiu que o próximo fim de semana seria palco de seu "Churrasco Lendário". Com uma lista de convidados que crescia a cada hora, ele se dirigiu ao seu açougue de confiança com uma missão: comprar as melhores carnes da cidade.

Empolgado, Marcelo não se conteve. Comprou picanha, maminha, linguiça toscana, coxa de frango, e até umas costeletas de porco. No total, foram N pacotes de carne, numerados de 1 a N, com o i-ésimo pacote tendo peso P_i.

O problema começou quando ele chegou ao caixa. Ele percebeu que o açougue cobrava por cada sacola usada, e portanto ele deseja minimizar o número de sacolas usadas. Porém, ele viu que cada sacola suporta no máximo peso S. Acima disso, a sacola pode rasgar.

Agora, Marcelo se vê diante de um dilema. Ele precisa levar todos os N pacotes de carne para casa, mas para agilizar o empacotamento, precisa organizar os pacotes em intervalos consecutivos, da forma como foram passados no caixa. Ou seja, cada sacola deve conter um intervalo contíguo de pacotes de carne. Mais precisamente:

- Para cada sacola, deve existir um par de inteiros (l, r) com $1 \leq l \leq r \leq N$ tal que a sacola contém os pacotes $l, l+1, \ldots, r$.
- A soma dos pesos em cada sacola deve ser no máximo S, ou seja, $P_l + P_{l+1} + ... + P_r \leq S$.
- Todo pacote deve estar em alguma sacola.
- Nenhum pacote deve estar em mais de uma sacola.

Ajude Marcelo a descobrir qual o número mínimo de sacolas que ele precisará usar, garantindo que a soma dos pesos em cada sacola não ultrapasse S.

Entrada

A primeira linha da entrada contém dois inteiros, N e S, que indicam o número total de pacotes de carne e o limite de peso que uma sacola suporta, respectivamente.

A segunda linha da entrada contém N inteiros, $P_1,\ P_2,\ \ldots,\ P_N$, que representam os pesos de cada pacote.

Saída

Seu programa deve imprimir uma única linha contendo um único inteiro, o número mínimo de sacolas que Marcelo precisa comprar.

Restrições

- $1 \leq N \leq 100\,000$.
- $1 \leq S \leq 1\,000\,000\,000$.
- $1 \leq P_i \leq S$ 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 (25 pontos):** $P_i = 1$ para todo $1 \leq i \leq N$.
- **Subtarefa 3 (25 pontos):** É garantido que Marcelo precisa de no máximo 3 sacolas.
- **Subtarefa 4 (25 pontos):** $N \leq 10$.
- **Subtarefa 5 (25 pontos):** Sem restrições adicionais.

Exemplos

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

Explicação do exemplo 1: Nesse caso o número mínimo de sacolas é 3. Podemos dividir os pacotes nas sacolas da seguinte maneira:

- A primeira sacola fica com o intervalo (1, 1), note que $2 \leq 6$.
- A segunda sacola fica com o intervalo (2, 3), note que $3 + 3 \leq 6$.
- A terceira sacola fica com o intervalo (4, 6), note que $1 + 1 + 2 \leq 6$.

Perceba que esta não é a única forma de dividir os pacotes nas sacolas. Note também que não podemos dividir em 2 sacolas da seguinte forma: $3 + 3 \leq 6$ e $2 + 1 + 1 + 2 \leq 6$ pois, apesar das sacolas suportarem os pesos, esta divisão não organiza os pacotes em intervalos consecutivos.

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

Exemplo de entrada 3	**Exemplo de saída 3**
8 10 6 3 4 5 2 7 3 5	4

Fotos de Relíquias

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

Jonas, um famoso arqueólogo, quer viralizar nas redes sociais publicando fotos de relíquias encontradas em suas jornadas. Para tirar fotos agradáveis, Jonas escolheu uma paisagem como plano de fundo para suas fotos.

A paisagem escolhida por Jonas é dividida em N seções, de forma que algumas são seções de destaque e outras não. Ele escreveu uma lista A de N inteiros que descrevem as N seções da paisagem, da esquerda para a direita, de forma que, se $A_i = 1$ então a i-ésima seção é de destaque, e caso contrário $A_i = 0$.

Jonas escolheu uma relíquia e vai tirar uma foto dela usando uma região contínua da paisagem como plano de fundo. Para que a relíquia seja o foco principal, a foto deve conter uma única seção de destaque, onde a relíquia será posicionada. Portanto, ele deve escolher um par de inteiros (l, r) que satisfaça as seguintes condições:

- $1 \leq l \leq r \leq N$.
- As seções l, $l + 1$, ..., r farão parte da foto.
- Dentre as seções presentes na foto, exatamente uma é de destaque, ou seja, existe um único índice k tal que $l \leq k \leq r$ e $A_k = 1$.

Jonas quer saber de quantas maneiras diferentes ele pode tirar a foto. Ajude ele a determinar a quantidade de pares de inteiros (l, r) que satisfazem todas as condições desejadas.

Entrada

A primeira linha da entrada contém um único inteiro N, indicando o número de seções na paisagem.

A segunda linha da entrada possui N inteiros A_1, $A_2, \ldots,$ A_N, onde $A_i = 1$ se a i-ésima seção é uma seção de destaque, ou $A_i = 0$ caso contrário.

Saída

Seu programa deve imprimir uma única linha contendo um único inteiro, o número de modos que Jonas pode tirar a foto.

Restrições

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

- $1 \leq N \leq 100\,000$.
- $A_i = 0$ ou $A_i = 1$ para todo $1 \leq i \leq N$.

Para competidores que utilizam C++ ou Java: Observe que alguns valores na saída podem ser muito grandes para caberem em um inteiro de 32 bits. É recomendado o uso de inteiros de 64 bits (`long long` em C++; `long` em Java). *(Competidores usando Python ou JavaScript podem ignorar este aviso.)*

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):** $N \leq 100$.
- **Subtarefa 3 (22 pontos):** $N \leq 3\,000$.
- **Subtarefa 4 (26 pontos):** Existe exatamente uma seção de destaque na paisagem, ou seja, há um único índice i na lista tal que $A_i = 1$.
- **Subtarefa 6 (32 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
5 1 0 0 1 0	9

Explicação do exemplo 1: Os pares (l, r) que seguem as condições para a foto são: $(1, 1)$, $(1, 2)$, $(1, 3)$, $(2, 4)$, $(2, 5)$, $(3, 4)$, $(3, 5)$, $(4, 4)$ e $(4, 5)$.

Exemplo de entrada 2	**Exemplo de saída 2**
12 0 0 0 1 0 0 0 0 0 0 0 0	36

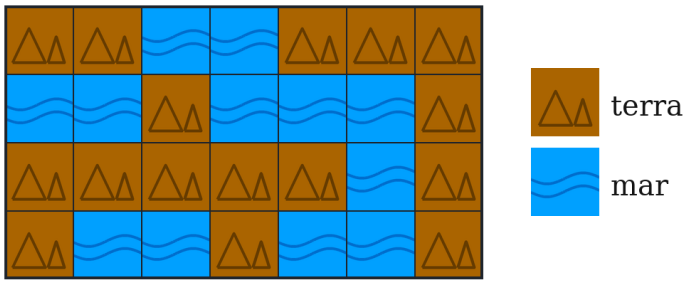
Explicação do exemplo 2: Este exemplo satisfaz as restrições da subtarefa 4.

Fitas Verde-amarelas

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

Com a ajuda dos alunos de Ciência da Computação, o Instituto de Geociências da Unicamp descobriu a existência de um arquipélago (conjunto de ilhas) no Oceano Atlântico. Para homenagear os alunos, o arquipélago foi chamado de Nlogônia. Agora, a professora Ada embarcou em uma viagem para visitar a Nlogônia pela primeira vez.

Ciente de que seu celular pode não ter sinal nas ilhas, Ada está levando um mapa que ela mesma desenhou. Este mapa consiste de uma malha quadriculada (*grid*) com N linhas e M colunas, onde cada célula pode ser terra ou mar. A figura abaixo ilustra um exemplo de mapa.

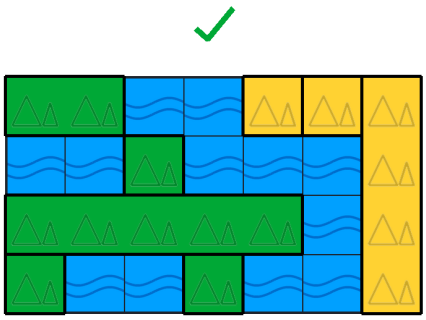


Com as turbulências do voo, o mapa acabou sendo danificado com rasgos e furos. Ada possui dois rolos de fita adesiva semi-transparente, um de fita verde e um de fita amarela, dos quais ela vai cortar fitas para consertar o mapa. Cada fita pode ser usada para cobrir um segmento contíguo (horizontal ou vertical) de células. Ao colar uma fita, Ada precisa alinhar a fita com as linhas da malha, de modo que cada célula esteja completamente coberta pela fita, ou completamente descoberta.

Ada consegue cortar fitas de qualquer comprimento que desejar, inclusive de comprimento 1 (que cobrem apenas uma célula). Além disso, ela pode cortar quantas fitas desejar de cada rolo, e confirmou que cada um dos rolos possui fita suficiente para cobrir o mapa inteiro se necessário. Porém, considerando a utilidade do mapa e o senso estético de Ada, ela estabeleceu as seguintes regras para consertar o mapa:

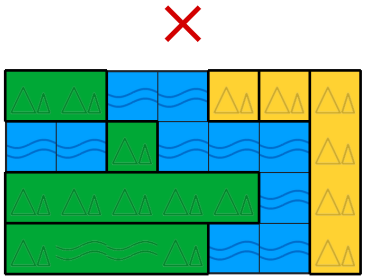
- Todas as células de terra devem estar cobertas por fitas.
- Todas as células de mar devem estar descobertas.
- Fitas verdes só podem ser coladas na horizontal. Ou seja, uma fita verde pode cobrir qualquer segmento contíguo de células na mesma linha.
- Fitas amarelas só podem ser coladas na vertical. Ou seja, uma fita amarela pode cobrir qualquer segmento contíguo de células na mesma coluna.
- Uma célula coberta com uma fita amarela **não** pode ser vizinha de uma célula coberta com uma fita verde. Duas células são vizinhas se elas compartilham um lado (ou seja, consideramos apenas vizinhas horizontais e verticais, não diagonais).

A figura abaixo ilustra um modo válido de consertar o mapa mostrado acima usando 8 fitas, sendo 5 verdes e 3 amarelas. Observe que é permitido que células vizinhas na diagonal sejam cobertas com a mesma cor.

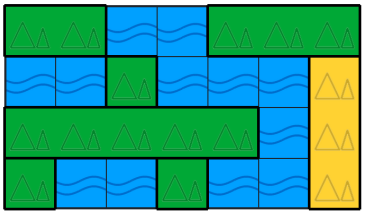


Configuração válida.

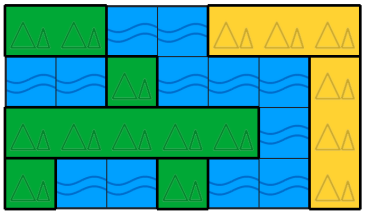
As figuras abaixo ilustram três modos inválidos de consertar o mapa usando 7 fitas. Em cada uma delas, alguma regra não é respeitada.



Existem células de mar que estão cobertas.



Existe uma fita verde adjacente a uma fita amarela.



Existe uma fita amarela colada na horizontal.

Apesar de toda a sua genialidade, a professora Ada tem dificuldade com o processo de cortar fitas dos rolos (ela reclama que as fitas colam nas mãos antes de ela usar a tesoura). Por isso, ela pediu sua ajuda. Ajude Ada a determinar o número **mínimo de fitas** que ela precisa colar no mapa para cobrir o mapa de maneira válida, ou seja, respeitando todas as regras acima.

Entrada

A primeira linha da entrada contém dois inteiros N e M, o número de linhas e o número de colunas do mapa, respectivamente.

As próximas N linhas possuem M caracteres cada (sem espaços em branco) e descrevem o mapa. O j-ésimo caractere da i-ésima linha, c_{ij}, é `'#'` (sem aspas) caso a célula (i, j) seja de terra, ou `'.'` (sem aspas) caso a célula (i, j) seja de mar.

Saída

Seu programa deve imprimir uma única linha contendo um único inteiro, o número mínimo de fitas que Ada precisa colar para consertar o mapa de acordo com as regras que ela estabeleceu.

Restrições

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

- $1 \leq N \leq 1\,000$.
- $1 \leq M \leq 1\,000$.
- $c_{ij} =$ `'#'` ou $c_{ij} =$ `'.'` para todos $1 \leq i \leq N$ 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 (10 pontos):** $N = 2$ e $M = 2$.
- **Subtarefa 3 (19 pontos):** $N = 1$.
- **Subtarefa 4 (12 pontos):**
  - $N \geq 2$ e $M \geq 2$.
  - Existe apenas uma célula de mar.
- **Subtarefa 5 (20 pontos):** As células de terra formam um retângulo sólido (formalmente, existem duas células (l_1, c_1) e (l_2, c_2) tais que uma célula (i, j) possui terra se, e somente se, $l_1 \leq i \leq l_2$ e $c_1 \leq j \leq c_2$).
- **Subtarefa 6 (19 pontos):**
  - $N \leq 100$ e $M \leq 100$.
  - Em cada linha e em cada coluna, as células de terra formam um segmento contíguo (ou seja, não existem dois segmentos disjuntos de terra na mesma linha ou na mesma coluna).
- **Subtarefa 7 (20 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
4 7 ##..### ..#...# #####.# #..#..#	8

Explicação do exemplo 1: Este é o exemplo mostrado no enunciado. O enunciado mostra um modo de cobrir o mapa usando oito fitas, e é possível provar que não há como cobrir o mapa de maneira válida usando sete ou menos fitas.

Exemplo de entrada 2	**Exemplo de saída 2**
1 9 ...####.#	2

Explicação do exemplo 2: Este exemplo satisfaz as restrições da subtarefa 3. Podemos usar duas fitas verdes para cobrir o mapa corretamente.

Exemplo de entrada 3	**Exemplo de saída 3**
6 5 ....### ..### ..### ..###	3

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

Exemplo de entrada 4	**Exemplo de saída 4**
8 9 ....##... ..####... ...###... ......##. ......### ....# ....# ##.......	7

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

Exemplo de entrada 5	**Exemplo de saída 5**
5 15 ####..####..### #..#..#..#...#. #..#..####...#. #..#..#..#...#. ####..####..###	18

Empregos de Júlio

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

Júlio trabalha muito para sustentar sua família. No entanto, ele percebeu que um único emprego não seria suficiente para pagar as contas e, por isso, decidiu arranjar dois! Assim, ele foi contratado para ser entregador de jornais e segurança de shopping. A princípio, ele pensou que conseguiria trabalhar em ambos ao mesmo tempo, mas logo descobriu que isso não seria tão fácil.

Como os dois serviços demandam muito tempo e esforço, Júlio desistiu de pegar turnos em ambos ao mesmo tempo. Dessa forma, ele pretende escolher em qual deles ele vai trabalhar em cada um dos próximos N dias, sendo que, para cada um desses, Júlio sabe o quanto ele ganharia se trabalhasse como entregador ou segurança. Mais especificamente, no i-ésimo dia, ele ganhará a_i reais para entregar jornais e b_i reais para vigiar o shopping.

O chefe de Júlio no emprego de segurança ficou sabendo da decisão de seu empregado e, a fim de segurá-lo no trabalho, ofereceu um incentivo a ele. Assim, ele prometeu a Júlio que, após trabalhar K dias na segurança do shopping, ele ganhará em dobro em todo e qualquer turno que ele escolher, ou seja, ganhará $2b_i$ reais se vigiar o shopping no i-ésimo dia. Note que os primeiros K dias trabalhados nesse emprego não precisam ser consecutivos, e ele ganhará o valor normal do dia.

Júlio está muito interessado na proposta do chefe, mas, como a quantidades N e K de dias são muito grandes, ele não sabe em qual emprego ele deve trabalhar em cada dia. Sua tarefa é: dados os valores N e K, bem como as listas $a_1 \dots a_N$ e $b_1 \dots b_N$, ajude Júlio a saber qual o **máximo** de dinheiro que ele pode conseguir, considerando a proposta feita por um de seus chefes.

Entrada

A primeira linha da entrada contém dois inteiros, N e K, que indicam o número de dias que Júlio irá trabalhar e a quantidade de vezes que ele deve trabalhar de segurança para ganhar em dobro.

A segunda linha da entrada contém N inteiros, $a_1, a_2, \dots, a_N$, que representam o quanto ele receberia em cada dia caso trabalhasse entregando jornais.

A terceira linha da entrada contém N inteiros, $b_1, b_2, \dots, b_N$, que representam o quanto ele receberia em cada dia caso trabalhasse vigiando o shopping.

Saída

A saída deve conter uma única linha contendo um inteiro, o máximo que Júlio consegue receber nos próximos N dias.

Restrições

- $1 \leq N \leq 100\,000$.
- $0 \leq K \leq N$.
- $1 \leq a_i, b_i \leq 1\,000\,000\,000$, para todo $1 \leq i \leq N$.

Para competidores que utilizam C++ ou Java: Observe que alguns valores na saída podem ser muito grandes para caberem em um inteiro de 32 bits. É recomendado o uso de inteiros de 64 bits (`long long` em C++; `long` em Java). *(Competidores usando Python ou JavaScript podem ignorar este aviso.)*

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 (13 pontos):** $K = 0$ e $N \leq 1\,000$.
- **Subtarefa 3 (21 pontos):** $K = 1$ e $N \leq 1\,000$.
- **Subtarefa 4 (31 pontos):** $N \leq 1\,000$.
- **Subtarefa 5 (14 pontos):** $b_i = b_j$, para todo $1 \leq i \leq j \leq N$.
- **Subtarefa 6 (21 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
5 0 30 40 30 50 70 10 30 20 20 40	260

Explicação do exemplo 1: Este exemplo satisfaz as restrições da subtarefa 2. Note que, como $K = 0$, Júlio ganha em dobro no segundo emprego desde o início. Assim, ele pode trabalhar como entregador nos dias 1 e 4, e como segurança nos dias 2, 3 e 5. Dessa forma, ele ganharia $30 + 2 \cdot 30 + 2 \cdot 20 + 50 + 2 \cdot 40 = 260$ reais.

Exemplo de entrada 2	**Exemplo de saída 2**
5 1 30 40 30 50 70 10 30 20 20 40	240

Explicação do exemplo 2: Este exemplo satisfaz as restrições da subtarefa 3. Note que, como $K = 1$, Júlio ganha em dobro no segundo emprego após o primeiro dia de trabalho como segurança. Assim, ele pode trabalhar como entregador no dia 4 e como segurança nos dias 1, 2, 3 e 5. Dessa forma, ele ganharia $10 + 2 \cdot 30 + 2 \cdot 20 + 50 + 2 \cdot 40 = 240$ reais.

Exemplo de entrada 3	**Exemplo de saída 3**
5 2 30 40 25 15 20 10 10 10 10 10	130

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