SAET 2024 - Maratona de Programação

29 de novembro de 2024

Instruções Importantes

- Use a opção **Runs** para enviar suas soluções. Os problemas podem resolvidos em qualquer ordem e linguagem (dentre C, C++ e Python, independentemente do problema);
- Suas soluções serão testadas com várias entradas, além das dada como exemplo. Por isso, sua solução pode não ser aceita mesmo se funcionar para os exemplos dados. Certifique-se que ela funciona para todas as entradas possíveis;
- A saída gerada deve ser *exatamente* conforme especificada. Em particular, **não** imprima instruções ("digite um número", "a resposta é", etc);
- É garantido que todas as entradas usadas para teste estarão de acordo com o enunciado, não sendo necessário testar se são válidas;
- Ao enviar uma solução, o sistema irá responder uma das seguintes respostas:
  - `Not answered yet`: a solução está sendo corrigida. Aguarde um pouco e atualize a página;
  - `YES`: solução aceita. Parabéns!
  - `Wrong Answer`: a saída impressa pelo seu programa não é a saída correta esperada, para alguma entrada de teste;
  - `Presentation Error`: a saída impressa está correta, exceto por espaços em branco e/ou quebras-de-linha faltando/sobrando;
  - `Time Limit Exceeded`: o tempo de execução do seu programa ultrapassou o tempo limite estipulado para o problema (ver tabela abaixo). O tempo de execução da sua solução precisa ser menor;
  - `Runtime Error`: seu programa gerou algum erro em tempo de execução ("crashou");
  - `Compile Error`: seu programa não compila.
- Todas as linhas, tanto na entrada quanto na saída, terminam com o caractere de fim-de-linha ($\backslash n$), mesmo quando houver apenas uma única linha na entrada e/ou saída;

- Sua solução deve processar cada arquivo de entrada no tempo máximo estipulado para cada problema, dado pela seguinte tabela:

Problema	**Nome**	**Tempo Limite (segundos)**
A	Alarme do Museu	1
B	Bolanacci	1
C	Cuca B.	1
D	Dupla do Crime	1
E	Escolhendo os Nomes	1
F	Fofoqueira	1
G	Geoguessr	1
H	Handshakes	1
I	Ilhas Toledanas	1
J	Jogo da Forca	1
K	Kummirub 2D++	1
L	Lagoa Azul	2
M	Montando a Dieta	1

A: Alarme do Museu

`Arquivo: alarme.[c|cpp|py]`

Um museu altamente seguro possui um corredor onde está exposta uma valiosa obra de arte. Para garantir a segurança, um sistema de alarme é acionado toda vez que um sensor de movimento detecta alguém no corredor.

Cada vez que o sensor detecta movimento, o sistema de alarme é ativado e permanece ligado por D segundos. Se uma nova detecção ocorrer antes que o sistema se desative, o tempo de ativação é reiniciado.

Dado o momento de cada detecção e a duração da ativação do alarme após cada detecção, determine o tempo total em que o alarme permaneceu ativo.

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 1000$), representando o número de detecções de movimento.

A segunda linha contém N inteiros T_i ($0 \leq T_i \leq 10^4$), em ordem crescente, onde cada T_i representa o segundo em que o sensor detectou movimento.

A terceira linha contém um inteiro D ($1 \leq D \leq 10^4$), representando a duração em segundos que o alarme permanece ativo após cada detecção.

Saída

Imprima o número total de segundos em que o alarme permaneceu ativo.

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

Exemplo de entrada	**Exemplo de saída**
5 1 3 6 10 15 5	19

Obs: no primeiro exemplo de entrada, o alarme ficou ativo nos intervalos de tempo $[1, 2]$ e $[4, 5, 6]$, totalizando 5 segundos.

B: Bolanacci

`Arquivo: bolanacci.[c|cpp|py]`

Em um reino distante, há uma antiga lenda sobre a sequência mística chamada Bolanacci. Conta-se que o grande mago Bolan desenvolveu uma fórmula secreta para prever eventos futuros, calculando a soma de energias mágicas do passado. Diz a lenda que essa sequência é usada para prever os momentos de maior poder do reino, e que só os estudiosos mais habilidosos conseguem entendê-la.

A sequência começa com três números iniciais. Cada número a partir do quarto é calculado somando as duas energias anteriores e subtraindo a terceira energia mais antiga. Somente os grandes matemágicos podem encontrar o termo correto na sequência para prever os dias de fortuna.

Dada a posição n, sua tarefa é determinar o enésimo termo da sequência Bolanacci e ajudar os estudiosos do reino a prever o futuro!

A sequência é definida da seguinte forma:

$$\begin{cases} B_1 = 1, \\ B_2 = 2, \\ B_3 = 4, \\ B_n = B_{n-1} + B_{n-2} - B_{n-3}, \quad \text{para } n > 3. \end{cases}$$

Entrada

A entrada consiste em um único inteiro n ($1 \leq n \leq 100$), representando a posição do termo que você deve encontrar na sequência Bolanacci.

É garantido que o valor do enésimo termo pedido estará dentro dos limites de um inteiro de 32 bits.

Saída

Imprima o enésimo termo da sequência Bolanacci.

Exemplo de entrada	**Exemplo de saída**
`4`	`5`

Exemplo de entrada	**Exemplo de saída**
`6`	`8`

Exemplo de entrada	**Exemplo de saída**
`8`	`11`

C: Cuca B.

`Arquivo: cuca.[c|cpp|py]`

Uma coisa que a quinta série adora fazer é brincar com nomes que soam como palavrões. Você certamente já viu a brincadeira de juntar um nome e um sobrenome que, quando lidos juntos, soam como um palavrão! A brincadeira é especialmente comum em programas ao vivo de televisão e de internet ("Abraço para o meu tio Cuca ...", "Beijo para minha tia Paula ...").

Dada uma lista de nomes e uma lista de sobrenomes, de quantas maneiras é possível escolher um sufixo (não vazio) de um nome e um prefixo (não vazio) de um sobrenome de forma que, juntos, foram um palavrão dado?

Por exemplo, considere o palavrão *boco*, os nomes *lobo* e *bob*, e os sobrenomes *costa*, *ocorvo* e *comadre*. Existem três maneiras de formar o palavão com as condições dadas: *lobo* + *costa*, *lobo* + *comadre* e *bob* + *ocorvo*. Em outro exemplo, considere o palavrão *aaa*, os nomes *aa* e *a*, e os sobrenomes *aa* e *aaa*. Há seis maneiras de formar o palavrão com as condições dadas: *aa* + *aa*, *aa* + *aa*, *aa* + *aaa*, *aa* + *aaa*, *a* + *aa* e *a* + *aaa*.

Entrada

A primeira linha contém uma string P ($2 \leq |P| \leq 10$) indicando o palavrão, contendo apenas letras minusculas. A próxima linha contém um inteiro N ($1 \leq N \leq 10^5$), o número de nomes. As próximas N linhas contém um nome cada, de até 10 letras minusculas. A próxima linha contém um inteiro M ($1 \leq M \leq 10^5$), o número de sobrenomes. As próximas M linhas contém um sobrenome cada, de até 10 letras minusculas.

Saída

Imprima de quantas maneiras é possível formar o palavrão com as condições dadas.

Exemplo de entrada	**Exemplo de saída**
boco 2 lobo bob 3 costa ocorvo comadre	3

Exemplo de entrada	**Exemplo de saída**
aaa 2 aa a 2 aa aaa	6

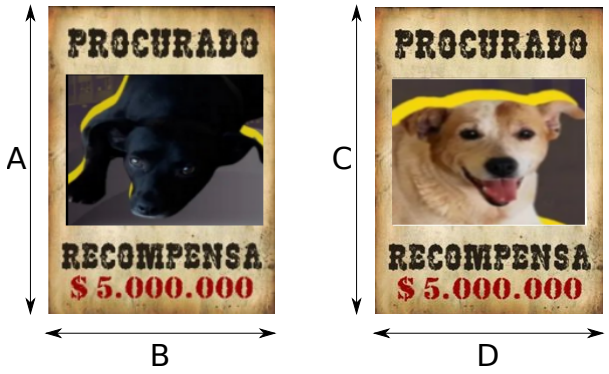
D: Dupla do Crime

`Arquivo: dupla.[c|cpp|py]`

Kubitschek e Chekskubit são os dois cachorros vira-latas que compõem a dupla de notórios criminosos procurados pelos seus terríveis, impiedosos, horripilantes, amedrontadores e imperdoáveis crimes cometidos nos arredores da UFOPR (Universidade Fictícia do Oeste do Paraná). Entre os crimes realizados pela dupla, o que mais tomou notoriedade foi o de cometer, mais de uma vez, o ato terrorista de comer restos das refeições dos alunos do campus.

Diante desse crime abominável, um dos setores da UFOPR, o RUF (Restaurante Universitário Fictício), entrou em contato com as autoridades e cobrou que tomassem providências contra esses bandidos perigosíssimos. Em resposta, a polícia iniciou uma campanha para capturá-los, imprimindo cartazes de "procurado" com recompensas para quem os entregasse.

Para tal, a polícia precisa saber qual deve as dimensões dos cartazes a serem impressos. A polícia definiu que ambos os cartazes devem ter a mesma proporção de altura/largura. Dadas a altura e largura do cartaz de "procurado" de Kubitschek e a altura do cartaz de "procurado" de Chekskubit, qual deve ser sua largura? Além disso, qual será a área total de cada cartaz?



Entrada

A entrada contém três inteiros A, B e C ($1 \leq A, B, C \leq 100$) indicando, em centímetros, a altura e largura do cartaz de "procurado" de Kubitschek e a altura do cartaz de "procurado" de Chekskubit, respectivamente.

Saída

Imprima uma linha com três valores D, A_K e A_C, sendo D a largura do cartaz de "procurado" de Chekskubit (em centímetros), A_K a área do cartaz de "procurado" de Kubitschek e A_C a área do cartaz de "procurado" de Chekskubit (em centímetros quadrados). Imprima todos os valores com duas casas decimais.

Exemplo de entrada	**Exemplo de saída**
`4 2 10`	`5.00 8.00 50.00`

Exemplo de entrada	**Exemplo de saída**
`3 1 10`	`3.33 3.00 33.33`

E: Escolhendo os Nomes

`Arquivo: escolhendo.[c|cpp|py]`

Você reparou que nesta prova o nome de todos os problemas começa com a mesma letra do problema? O problema A (*Alarme do Museu*) começa com a letra A; o problema B (*Bolanacci*) começa com a letra B; e assim por diante! Nós fizemos isso para que a prova fique mais divertida!

Nem sempre é fácil escolher o nome de um problema. Por exemplo, o próximo problema desta prova começou se chamando *Bisbilhoteira*, depois virou *Enxerida* e finalmente entrou na prova como *Fofoqueira*. O hotel do problema I (*Ilhas Toledanas*), por exemplo, só tem esse nome porque precisávamos de algo iniciando com a letra I.

Já estamos montando nossa próxima prova, e precisamos da sua ajuda para escolher os nomes dos problemas dela. Temos N problemas disponíveis para uso no nosso banco de problemas. Cada problema tem uma lista de possíveis nomes que ele pode ter. A prova deve ser montada de forma que:

- O nome de um problema deve começar com a mesma letra do problema;
- Seguindo a ordem alfabética, não pode haver "buracos" nas letras dos problemas. Em outras palavras, o problema B não pode existir se não existir o problema A; o problema C não pode existir se não existir o problema B; o problema D não pode existir se não existir o problema C; e assim por diante.

Considerando as restrições dadas, qual é o número máximo de problemas que nossa próxima prova poderá ter?

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 100$), o número de problemas disponíveis. As próximas N linhas descrevem um problema cada. Cada linha é dada por um inteiro K ($1 \leq K \leq 30$) indicando o número de possíveis nomes do problema, seguido pelos K nomes. Os nomes terão no máximo 10 caracteres, sendo o primeiro uma letra maiúscula e os demais letras minúsculas.

Saída

Imprima uma linha com um inteiro indicando o número máximo de problemas que a prova pode ter.

Exemplo de entrada	**Exemplo de saída**
6 2 Bolacha Cookie 1 Carros 2 Alarme Detonacao 1 Biscoito 3 Festa Encontro Festival 2 Festival Evento	3

F: Fofoqueira

`Arquivo: fofoqueira.[c|cpp|py]`

Nossa amiga Ana está enfrentando um problema curioso. Ela tem uma irmã bisbilhoteira que insiste em ler o seu diário sempre que tem a oportunidade, para depois ir fofocar sobre o que leu para suas amigas. Para evitar que seus segredos sejam descobertos, Ana começou a codificar suas mensagens, repetindo os caracteres para deixar as frases mais confusas. Por exemplo, em vez de escrever "amor", ela pode escrever "aaammmoorrrr".

No entanto, agora ela está com outro problema: as mensagens ficaram muito grandes, ocupando páginas inteiras do diário. Para ajudar Ana, precisamos criar um algoritmo que compacte essas mensagens, substituindo as sequências de caracteres repetidos pelo caractere seguido pelo seu numero de repetições.

Entrada

A entrada consiste de uma string s ($1 \leq |s| \leq 10^5$), composta apenas por letras minusculas sem espaços.

Saída

Imprima uma linha contendo a string codificada.

Exemplo de entrada	**Exemplo de saída**
`aaaaabbccc`	`a5b2c3`

Exemplo de entrada	**Exemplo de saída**
`ppppccii`	`p4c2i2`

Exemplo de entrada	**Exemplo de saída**
`a`	`a1`

G: Geoguessr

`Arquivo: geoguessr.[c|cpp|py]`

No jogo *Geoguessr*, o computador escolhe um ponto secreto (x_S, y_S) no mapa mundi e mostra para cada um dos N jogadores algumas imagens deste ponto. Cada jogador então faz um palpite distinto (x_i, y_i), e vence aquele com a menor distância do seu palpite ao ponto secreto.

Os jogadores estão desconfiando que pode haver algum problema com o jogo, que pode estar apresentando distâncias inconsistentes ou ambíguas. Dados os palpites dos jogadores e suas distâncias para o ponto secreto, é possível determinar *com certeza* qual é o ponto secreto?

Entrada

A primeira linha contém um inteiro N ($3 \leq N \leq 10^5$), o número de jogadores. Cada uma das próximas N linhas contém três inteiros cada: x_i, y_i e R_i ($-10^7 \leq x_i, y_i \leq 10^7, 0 \leq R_i \leq 10^{16}$) indicando um palpite e o quadrado da distância para o ponto secreto (isto é, a distância de (x_i, y_i) para (x_S, y_S) é $\sqrt{R_i}$). É garantido que todos os palpites são distintos.

Saída

Se não for possível determinar com certeza o ponto secreto, imprima `impossivel`. Caso contrário, imprima os valores de x_S e y_S com duas casas decimais.

Exemplo de entrada	**Exemplo de saída**
3 5 2 10 2 2 1 -2 5 20	2.00 3.00

Exemplo de entrada	**Exemplo de saída**
3 0 0 2 1 -1 0 2 -2 2	1.00 -1.00

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

Exemplo de entrada	**Exemplo de saída**
3 4 3 16 6 3 9 12 3 36	impossivel

H: Handshakes

`Arquivo: handshakes.[c|cpp|py]`

Apertos de mão são comuns no cotidiano, seja para cumprimentar um conhecido, realizar uma negociação, ou até mesmo na política! Segundo a psicologia, na teoria dos seis graus de separação de Stanley Milgram, todas as pessoas no planeta estão separadas por no máximo seis apertos de mão. Você, como um desenvolvedor júnior para a *Handshakes*, famosa rede social no seu país, decidiu tentar pôr à prova essa teoria.

Dada uma lista de pessoas e de apertos de mãos já realizados, determine a menor distância (em apertos de mãos) entre duas pessoas.

Entrada

A primeira linha da entrada contém o número p de pessoas ($2 \leq p \leq 300$). As próximas p linhas possuem uma string S cada que representa o nome de cada pessoa ($1 \leq |S| \leq 20$; S não contém espaços).

Em seguida, há p linhas representando as conexões de cada pessoa. Cada linha contém o nome de uma pessoa seguido por uma lista de pessoas das quais ela já apertou a mão. A lista termina com -1.

Note que se A já apertou a mão de B, então B também já apertou a mão de A, mesmo que A não apareca na lista de apertos de B.

A última linha possui dois nomes M e N (dentre os anteriormente listados).

Saída

Imprima uma linha com a menor distância (em apertos de mãos) entre as pessoas de nomes M e N. Caso não haja um caminho entre elas, imprima -1.

Exemplo de entrada	Exemplo de saída
10 Alice Bob Carol Dave Eve Frank Grace Heidi Ivan Judy Alice Bob Carol -1 Bob Dave Eve -1 Carol Frank Grace -1 Dave Heidi -1 Eve Ivan -1 Frank Judy -1 Grace Heidi Ivan -1 Heidi Judy -1 Ivan -1 Judy -1 Dave Grace	2

Exemplo de entrada	Exemplo de saída
15 Alice Bob Carol Dave Eve Frank Grace Heidi Ivan Judy Kevin Lucy Mallory Nancy Oscar Alice Bob Carol -1 Bob Dave Eve -1 Carol Frank -1 Dave Grace -1 Eve Heidi -1 Frank Ivan -1 Grace Judy -1 Heidi Kevin -1 Ivan Lucy -1 Judy Mallory -1 Kevin Nancy -1 Lucy Oscar -1 Mallory -1 Nancy -1 Oscar -1 Carol Kevin	5

I: Ilhas Toledanas

`Arquivo: ilhas.[c|cpp|py]`

O grande hotel *Ilhas Toledanas Summer Resort* é um destino bastante procurado pelos turistas nesta época do ano! O hotel conta com restaurante, piscina, academia, SPA, cancha esportiva e, claro, seus N quartos luxuosos.

O gerente do hotel ficou muito animado ao perceber que o hotel já recebeu R reservas para a temporada. Entretanto, são tantas reservas que ele se perdeu um pouco. Por isso, ele pediu sua ajuda para, dada a lista de reservas, determinar qual o número máximo de quartos que estarão ocupados simultaneamente, e em qual dia isso ocorrerá.

Entrada

A primeira linha contém dois inteiros N e R ($1 \leq N, R \leq 10^5$), o número de quartos e de reservas. Considere que os quartos são numerados de 1 a N. As próximas R linhas contém uma reserva cada. Cada linha contém três inteiros Q, A e B ($1 \leq Q \leq N$, $1 \leq A \leq B \leq 10^9$), indicando que o quarto Q está reservado entre os dias A e B (inclusive nos dias A e B, durante todo o dia). É garantido que não há sobreposição de reservas de um mesmo quarto.

Saída

Imprima uma linha com dois inteiros M e D, indicando o número máximo de quartos ocupados simultaneamente e em que dia esta ocupação ocorrerá. Se a ocupação máxima ocorrer em mais um dia, imprima o primeiro dia no qual haverá esta ocupação.

Exemplo de entrada	**Exemplo de saída**
3 4 1 2 5 3 6 6 1 7 10 2 5 8	2 5

Exemplo de entrada	**Exemplo de saída**
3 6 2 8 20 3 2 3 1 10 15 3 6 12 1 1 5 2 4 7	3 10

J: Jogo da Forca

`Arquivo: jogo_da_forca.[c|cpp|py]`

João e seu irmão Miguel querem brincar de jogo da forca, porém o papel da casa deles acabou. João então teve a ideia de implementar um jogo da forca utilizando suas habilidades de programação, porém seu programa *crashou* no primeiro "Hello World". João então pediu para que você implemente o jogo da forca em seu lugar.

Entrada

A primeira linha da entrada contém a palavra P ($1 \leq |P| \leq 100$; P contém apenas letras minúsculas) a ser adivinhada. As próximas linhas contém uma letra minuscula c cada, indicando um palpite. A entrada termina com `*`.

Saída

Após cada palpite, imprima uma linha o progresso em acertar a palavra. Para isso, utilize o caractere *underline* (`_`) para representar as letras da palavra que ainda não foram descobertas, conforme o formato dado nos exemplos abaixo.

Exemplo de Entrada	**Exemplo de Saída**
maratona a c e t o p s s t *	_a_a___a _a_a___a _a_a___a _a_at__a _a_ato_a _a_ato_a _a_ato_a _a_ato_a _a_ato_a

Exemplo de Entrada	**Exemplo de Saída**
toledo a t o l e o d u *	______ t_____ to___o tol__o tole_o tole_o toledo toledo

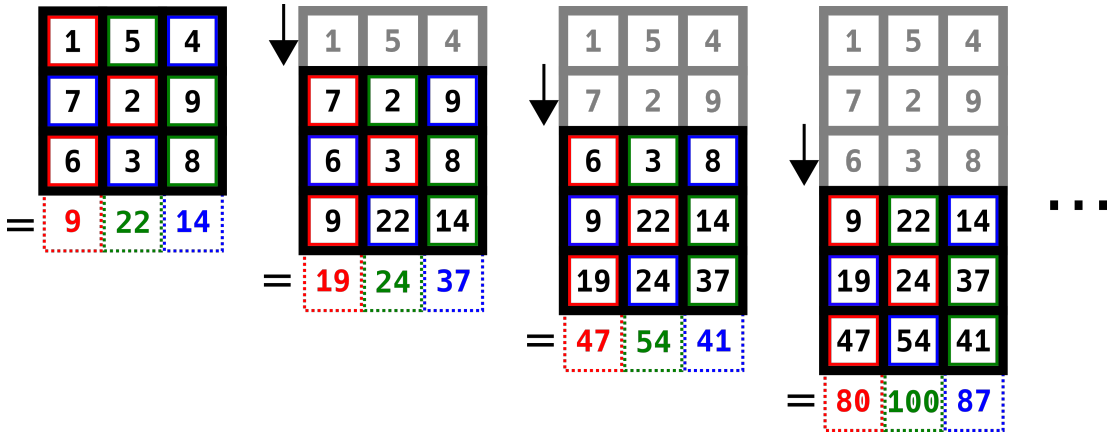
K: Kummirub 2D++

`Arquivo: kummirub.[c|cpp|py]`

A criação de novos jogos de tabuleiro da UT Fábrica de Passatempos Recreativos está buscando por estagiários criativos, que possam atualizar o portfólio de produtos da empresa. Liana e Hipóleto mal entraram no programa de estágio, e já deram um abacaxi para eles resolverem.

Por gostarem de xadrez, os recrutadores jogaram a dupla no setor de tabuleiros. Acontece que esse departamento não só está muito atrás das tendências do mundo dos jogos, como as criações elaboradas têm sido cada vez mais "racha-cucas", por assim dizer. Qualquer um que quiser entender o porquê, basta ver o manual do jogo *Kummirub: 2D!*, que foi apresentado para Liana e Hipóleto:

> **Kummirub: 2D!**
>
> *Classificação: 10+*
>
> *De 2 a 10 jogadores.*
>
> Agora o **Kummirub** virou **2D!** Em uma nova versão, este jogo irá desafiar as suas capacidades de raciocínio, e **apenas gênios poderão vencer!**
>
> **Como jogar**: Para N jogadores, os números de 1 a N^2 devem ser dispostos aleatoriamente nas casas do tabuleiro $N \times N$. No começo do jogo, cada jogador escolherá sua posição i de 1 a N. Em seguida, cada um deve escolher uma casa para cada linha do tabuleiro, totalizando N casas para cada. Essas casas não poderão ser trocadas até o final da partida, e uma posição não pode pertencer a dois jogadores numa mesma rodada.
>
> Então, uma nova linha do tabuleiro será formada. Cada jogador deve somar os números das suas casas escolhidas, e colocar o resultado na sua posição: a i-ésima casa da nova linha.
>
> Por fim, as casas escolhidas pelos jogadores devem descer uma linha, e uma nova rodada é iniciada. Ganha o jogador que não cometer erros ao realizar as operações de cada rodada!
>
> Um exemplo de jogo com três jogadores pode ser visto a seguir. Para ilustrar as escolhas dos jogadores, o jogador 1 é o vermelho, o jogador 2 é o verde, e o jogador 3 é o azul:



Liana e Hipóleto devem desenvolver uma calculadora que calcule os resultados após o T-ésimo turno. Você ficou responsável pelo código, e eles pelo protótipo do dispositivo, que será incluso na próxima versão do passatempo: *Kummirub: 2D++!*.

Está na hora de colocar suas habilidades de maratonista em prática! Esta primeira versão do software deverá apresentar os resultados módulo $10^9 + 7$.

Entrada

A primeira linha contém o número de jogadores N ($2 \leq N \leq 10$).

As próximas N linhas correspondem ao tabuleiro. Cada linha contém N números $C_{x,y}$ ($1 \leq C_{x,y} \leq N^2$, $0 \leq x, y < N$), o número na casa de posição (x, y).

A linha seguinte contém o número do turno T ($1 \leq T \leq 10^8$).

Por fim, as N linhas seguintes são referentes às casas escolhidas pelos jogadores. Cada linha contém N números $i_{x,y}$ ($1 \leq i \leq N$), o número do jogador que escolheu a posição (x, y).

Saída

Imprima uma linha com N números r_i: o resultado do jogador i após T turnos, em módulo $10^9 + 7$.

Não coloque um espaço após o último resultado, apenas uma nova linha ('\n').

Exemplo de entrada	**Exemplo de saída**
3 1 5 4 7 2 9 6 3 8 4 1 2 3 3 1 2 1 3 2	80 100 87

Exemplo de entrada	**Exemplo de saída**
2 1 2 3 4 10000 2 1 1 2	492026538 492026538

L: Lagoa Azul

`Arquivo: lagoa_azul.[c|cpp|py]`

Em um país, há várias regiões monitoradas pela empresa Lagoa Azul, que registra o volume de chuvas. Em cada região, há uma caixa coletora c que acumula o volume de chuva que cai sobre ela. Os dados dessas caixas ajudam no monitoramento e controle de enchentes, no planejamento agrícola, e na análise de eventos climáticos extremos. As caixas estão numeradas de 1 até n, sendo n o número total de caixas coletoras no país.

Você foi encarregado de implementar um sistema que permita:

- **Adicionar volume de chuva** a uma caixa específica.
- **Consultar o volume total acumulado de chuva** em um intervalo de caixas, de modo a analisar o total de chuvas acumuladas em uma determinada faixa de regiões.

Entrada

A primeira linha da entrada contém dois inteiros, n e q, onde:

- n ($1 \leq n \leq 10^5$) é o número de caixas coletoras.
- q ($1 \leq q \leq 10^5$) é o número de operações a serem realizadas.

A segunda linha contém n inteiros $v_1, v_2, \ldots, v_N$, onde v_i ($0 \leq v_i \leq 10^6$) é o volume inicial de chuva acumulado na caixa coletora i.

As próximas q linhas descrevem as operações, que podem ser de dois tipos:

- $1\ x\ v$: onde 1 representa uma operação de adição de chuva. Adicione v ($-100 \leq v \leq 100$) ao volume de chuva acumulado na caixa x ($1 \leq x \leq n$).
- $2\ l\ r$: onde 2 representa uma operação de consulta de volume acumulado. Consulte o volume total de chuva acumulada nas caixas de l a r ($1 \leq l \leq r \leq n$) inclusive.

Por exemplo, considerando um caso em que existem apenas 5 caixas d'águas, contendo o volume 1, 2, 3, 4, 5, nesta ordem.

- **Operação 1** (`2 1 3`): Consulta do volume total das caixas 1 a 3.
  - Caixas 1, 2 e 3 têm volumes iniciais de $1, 2,$ e 3, respectivamente.
  - **Resultado**: $1 + 2 + 3 = 6$
- **Operação 2** (`1 2 10`): Atualização do volume da caixa 2, adicionando 10.
  - O volume da caixa 2 passa de 2 para 12.
  - Novo estado das caixas: $[1, 12, 3, 4, 5]$.

- **Operação 3** (`2 1 3`): Consulta do volume total das caixas 1 a 3 após a atualização.
  - Caixas 1, 2 e 3 têm agora volumes de $1, 12,$ e 3.
  - **Resultado**: $1 + 12 + 3 = 14$
- **Operação 4** (`2 3 5`): Consulta do volume total das caixas 3 a 5.
  - Caixas 3, 4 e 5 têm volumes de $3, 4,$ e 5.
  - **Resultado**: $3 + 4 + 5 = 12$
- **Operação 5** (`2 1 5`): Consulta do volume total das caixas 1 a 5.
  - Caixas 1 a 5 têm volumes de $1, 12, 3, 4,$ e 5.
  - **Resultado**: $1 + 12 + 3 + 4 + 5 = 25$

Note que o volume de chuvas pode se tornar negativo durante as operações.

Saída

Para cada operação do tipo 2, imprima uma linha contendo o volume total de chuva acumulado nas caixas de l a r inclusive.

Exemplo de entrada	**Exemplo de saída**

`5 5`
`1 2 3 4 5`
`2 1 3`
`1 2 10`
`2 1 3`
`2 3 5`
`2 1 5`
	`6`
	`16`
	`12`
	`25`

Exemplo de entrada	**Exemplo de saída**

`5 5`
`0 0 0 0 0`
`1 1 10`
`1 3 5`
`1 5 7`
`2 1 5`
`2 3 5`
	`22`
	`12`

M: Montando a Dieta

`Arquivo: dieta.[c|cpp|py]`

Luquinhas está muito empenhado em "colocar o *shape*"! Ele vai treinar todos os dias, sem falta, na academia próxima da universidade. Sua meta é um dia conseguir um *shape* digno de um maromba de respeito!

Para cumprir sua meta, Luquinhas sabe que, além do treino, também precisa seguir uma dieta rigorosa. Seu nutricionista recomendou que ele consuma *no mínimo* C Calorias e *no máximo* G gramas de gordura por dia.

Luquinhas precisa agora fazer suas compras do dia. Há N alimentos disponíveis no supermercado. Luquinhas pode comprar no máximo uma unidade de cada alimento. Sabendo a quantidade de Calorias e de gordura contidas em cada alimento, qual será o valor mínimo total que Luquinhas deverá gastar para poder seguir sua dieta recomendada?

Entrada

A primeira linha contém os inteiros N, C e G ($1 \leq N \leq 100$, $1 \leq C \leq 2000$, $0 \leq G \leq 10$), o número de alimentos disponíveis, a quantidade mínima de Calorias e a quantidade máxima de gordura (em gramas), respectivamente.

Cada uma das próximas N linhas contém dois inteiros c_i e g_i ($0 \leq c_i, g_i \leq 10^9$), além de um valor real com duas casas decimais p_i ($0.00 \leq p_i \leq 10^5.00$) indicando a quantidade de Calorias e de gordura (em gramas) contidas no alimento, além do seu preço (em reais).

Saída

Se não for possível fazer as compras de forma a seguir a dieta recomendada, imprima `-1`. Caso contrário, imprima o valor mínimo que Luquinhas deve gastar, em reais, com duas casas decimais.

Exemplo de entrada	**Exemplo de saída**

`5 2000 10`
`455 1 0.72`
`300 3 2.95`
`500 2 5.80`
`1200 6 3.99`
`3500 11 0.10`
	`10.51`

Exemplo de entrada	**Exemplo de saída**

`3 1000 8`
`520 5 240.00`
`480 4 320.00`
`150 3 100.00`
	`-1`