

I Maratona Padawan de Programação do IFB

Caderno de Problemas

17 de outubro de 2026



(Este caderno contém 13 problemas)

Comissão Organizadora:

Arthur Andrade D'Olival

Caio Lauro de Lima

Daniel Saad Nogueira Nunes

Gabriel César Silvino Xavier

Igor de Oliveira Moura

João Carlos Gonçalves de Oliveira Filho

Orientações

- É permitido consultar livros, anotações ou qualquer outro material impresso durante a prova, entretanto, o mesmo não vale para materiais dispostos eletronicamente.
- A correção é automatizada, portanto, siga atentamente as exigências da tarefa quanto ao formato da entrada e saída conforme as amostras dos exemplos. Deve-se considerar entradas e saídas padrão;
- Para cada problema, além dos testes públicos, o juiz executará a sua submissão contra uma série de testes secretos para fornecer um parecer sobre a correção do programa.
- Procure resolver o problema de maneira eficiente. Se o tempo superar o limite pré-definido, a solução não é aceita. Lembre-se que as soluções são testadas com outras entradas além das apresentadas como exemplo dos problemas;
- Utilize a aba *clarification* para dúvidas da prova. Os juízes podem opcionalmente atendê-lo com respostas acessíveis a todos;

C/C++

- Seu programa deve retornar zero, executando, como último comando, `return 0` ou `exit(0)`.
- É sabido que em alguns casos de problemas com entradas muito grandes, os objetos `cin` podem ser lentos, por conta da sincronização de buffers da biblioteca `stdio`.
- Caso se deseje utilizar `cin` e `cout`, um jeito de se contornar este problema é executando-se o comando `ios_base::sync_with_stdio(0)`, no início de sua função `main`.
- Note que, neste caso, o uso de `scanf` e `printf` no mesmo programa é contra-indicado, uma vez que, com buffers separados, comportamentos inadequados podem ocorrer.

Java

- Não declare “package” no seu programa Java.
- Note que a convenção para o nome do arquivo fonte deve ser obedecida, o que significa que o nome de sua classe pública deve ser uma letra maiúscula (A, B, C, etc).
- Comando para executar uma solução Java:
`java -Xms1024m -Xmx1024m -Xss100m codigo_do_problema`

Kotlin

- Não declare “package” no seu programa Kotlin.
- Note que a convenção para o nome do arquivo de solução deve ser obedecida, o que significa que seu arquivo-fonte deve ser nomeado (A.kt, B.kt, C.kt, etc).
- Comando para executar uma solução Kotlin:
`java -Xms1024m -Xmx1024m -Xss100m codigo_do_problemaKt`

Python

- Note que apenas Python 3 é suportado.
- As submissões em Python terão sua sintaxe verificada; programas que não passarem nessa verificação receberão o veredito “Compilation Error”.

Problema A – Alpinismo

Autor: Daniel Saad Nogueira Nunes

Roberto Duran, o *mãos de pedra*, resolveu descansar um pouco de seu treinamento para a luta contra Sugar Ray Leonard e decidiu escalar a cordilheira Tamalanca. Essa cordilheira é composta por n montanhas, cada uma com uma altura v_i . Roberto quer saber quantos picos tem a cordilheira, ou seja, quantas montanhas são mais altas que as suas vizinhas imediatas. A primeira e a última montanha da cordilheira possuem apenas uma vizinha.

Ajude Duran a contar o número de picos da cordilheira.

Entrada

A primeira linha da entrada possui um inteiro n ($1 \leq n \leq 10^5$), representando o número de montanhas da cordilheira. A segunda linha possui n inteiros v_i ($1 \leq v_i \leq 10^9$), representando a altura de cada montanha.

Saída

Imprima uma linha com o número de picos da cordilheira.

Exemplo

Entrada	Saída
7 1 3 2 5 4 7 2	3
3 3 2 3	2
9 1 2 3 4 5 4 3 2 1	1

Problema B – Bodybuilder

Autor: Caio Lauro de Lima

João José pretende começar a treinar na academia, após muita insistência do seu irmão, Joaquim Joel, um famoso *bodybuilder*.

Joaquim montou um excelente plano de treino para seu irmão, contando com exercícios variados para os mais diversos grupos musculares. Na musculação, cada exercício é dividido em séries, que consistem em fazer um certo número de repetições, com um descanso entre uma série e a próxima. Por exemplo, “3 séries de 8 repetições” significa fazer 8 repetições, descansar, fazer mais 8, descansar e fazer outras 8.

João anda muito ocupado com sua vida acadêmica e quer economizar tempo para conseguir assistir aos episódios semanais da sua série favorita, “Circuito da Esfera de Aço”. Como não entende nada de musculação, decidiu fazer todas as repetições de cada exercício de uma só vez, sem descansar entre as séries. Para economizar ainda mais tempo, João gostaria da sua ajuda para calcular quantas repetições ele fará, no total, em cada exercício do seu plano de treino.

Entrada

A primeira linha da entrada contém um inteiro n ($1 \leq n \leq 100$), representando o número de exercícios no plano de treino do João. Cada uma das próximas n linhas possui dois inteiros s e r ($1 \leq s, r \leq 100$), separados por espaço, indicando o número de séries e o número de repetições por série do exercício, respectivamente.

Saída

Para cada um dos n exercícios, na mesma ordem da entrada, imprima uma linha com o número total de repetições que João fará nele.

Exemplo

Entrada	Saída
3	24
3 8	20
2 10	25
5 5	
2	20
10 2	24
6 4	
1	1
1 1	

Notas

No primeiro exemplo, temos 3 exercícios:

- No primeiro, temos 3 séries de 8 repetições, logo, $3 \cdot 8 = 24$ repetições;

- No segundo, 2 séries de 10 repetições, logo $2 \cdot 10 = 20$ repetições; e
- No terceiro, 5 séries de 5 repetições, logo $5 \cdot 5 = 25$ repetições.

Problema C – Criptografia

Autor: Daniel Saad Nogueira Nunes

Um dos passos na geração de chaves privadas e públicas no algoritmo RSA é a escolha de dois números primos grandes, p e q . A partir desses números, calcula-se $n = p \cdot q$ e realizam-se uma série de operações sobre n para gerar as chaves. A segurança do método depende da dificuldade de fatorar n em seus fatores primos p e q . Os números que são produtos de dois primos são conhecidos como **semiprimos**.

Você, como aspirante a criptógrafo, decidiu estudar a dificuldade de fatoração de números grandes e a distribuição de números semiprimos. Para isso, você quer escrever um programa que determine quantos números semiprimos existem em um intervalo $[l, r]$ dos naturais.

Entrada

A primeira linha da entrada descreve um inteiro n ($1 < n \leq 10^5$), que é o número de consultas a serem respondidas. As próximas n linhas, descrevem, cada uma, o intervalo da consulta por meio de dois inteiros, l e r ($1 \leq l \leq r \leq 10^7$).

Saída

Imprima, para cada consulta, o número de números semiprimos no intervalo $[l, r]$.

Exemplo

Entrada	Saída
3	4
1 10	3
10 20	4
20 30	

Notas

Existem 4 semiprimos no intervalo $[1, 10]$, que são: 4, 6, 9 e 10.

Existem 3 semiprimos no intervalo $[10, 20]$, que são: 10, 14 e 15.

Finalmente, existem 4 semiprimos no intervalo $[20, 30]$, que são: 21, 22, 25 e 26.

Problema D – Dança das Cadeiras

Autor: Igor de Oliveira Moura

Após uma Maratona de Programação do IFB, os participantes quiseram fazer uma brincadeira, a Dança das Cadeiras, assim, eles pegaram todas as cadeiras do campus, colocaram-nas no pátio e as posicionaram elas em um formato de um círculo. Cada participante se movimentava para a próxima cadeira, no sentido horário, a cada batida da música, porém, notaram que não colocaram uma cadeira a menos, como propõe a brincadeira original.

Descubra como ficou a configuração final dos participantes nas cadeiras após um determinado número de batidas da música.

Entrada

A primeira linha contém dois inteiros N e K ($1 \leq N \leq 3000$ e $0 \leq K \leq 3000$), representando, respectivamente, a quantidade de participantes e a quantidade de batidas da música.

A segunda linha contém N inteiros $a_1, a_2, \dots, a_N$ ($1 \leq a_i \leq N$), no qual o i -ésimo valor representa o identificador do participante que está a frente da i -ésima cadeira inicialmente.

Saída

Imprima N inteiros representando a configuração dos participantes após as K batidas da música.

Exemplo

Entrada	Saída
5 2	4 5 1 2 3
1 2 3 4 5	
4 1	3 2 1 4
2 1 4 3	
7 0	1 2 3 4 5 6 7
1 2 3 4 5 6 7	

Problema E – Estratégia de Pneus

Autor: Igor de Oliveira Moura

O chefe de equipe **Toto Saad** estava se preparando para uma importante corrida de Fórmula 1 e decidiu montar uma estratégia para os corredores do **IFB**.

Como pneus não são exatamente baratos, Toto quer definir uma estratégia que permita aos pilotos completar toda a corrida utilizando os conjuntos de pneus de forma eficiente.

O circuito possui N voltas. Para a corrida, existem três tipos de pneus, todos disponíveis em quantidade ilimitada:

- **Duro** (H): dura exatamente H voltas;
- **Médio** (M): dura exatamente M voltas;
- **Macio** (S): dura exatamente S voltas.

Toto é um chefe muito conservador e não gosta de desperdiçar pneus. Por isso, ele só aceita colocar um conjunto de pneus se a corrida ainda tiver voltas suficientes para esgotar a vida útil dele por completo. Entre os tipos que satisfazem essa condição, ele sempre prefere o que dura mais. Quando o número de voltas é insuficiente para esgotar o pneu duro ou o médio, Toto não tem escolha: usa o pneu macio.

Dados N , H , M e S , descubra qual estratégia Toto utilizará, informando a quantidade de conjuntos de pneus e a sequência de pneus utilizados, na ordem em que foram colocados.

Entrada

A entrada contém quatro inteiros N , H , M e S .

- N representa o número de voltas da corrida;
- H representa o número máximo de voltas suportadas por um conjunto de pneus duros;
- M representa o número máximo de voltas suportadas por um conjunto de pneus médios;
- S representa o número máximo de voltas suportadas por um conjunto de pneus macios.

É garantido que:

$$1 \leq N, H, M, S \leq 10^5$$

e:

$$H > M > S$$

Saída

A saída deve conter duas linhas.

Na primeira linha, imprima a **a quantidade de pneus** necessários para completar a corrida.

Na segunda linha, imprima a **sequência dos pneus utilizados, na ordem em que foram colocados durante a corrida**.

Exemplo

Entrada	Saída
10	2
8 4 2	H S
70	5
80 50 5	M S S S S
74	8
25 10 1	H H M M S S S S
7	7
12 9 1	S S S S S S S

Notas

No primeiro caso, temos 10 voltas

- O pneu duro tem 8 voltas de vida, logo, $10 - 8 = 2$ voltas restantes
- Como o pneu macio tem exatamente 2 voltas de vida, podemos usá-lo para completar as 10 voltas, em vez de usar um médio

Problema F – Foos Contra Fighters

Autor: Caio Lauro de Lima

Os *Foos* são criaturas pequenas que adoram invadir estúdios de música. Recentemente, eles invadiram o estúdio de música do IFB e esconderam os cabos, palhetas e outros equipamentos pequenos, bagunçando tudo.

Para devolvê-los ao seu habitat natural, foi criada a força-tarefa *Fighters*, que usa pequenos potes para capturá-los. Cada pote utilizado consegue transportar até um *Foo*. O líder da força-tarefa criou um programa que analisa as câmeras de segurança e imprime a sequência “foo” na tela sempre que detecta um *Foo*. Como foi feito às pressas, o programa também imprime letras soltas e sequências diferentes de foo, como “fo” ou “oof”, que, sozinhos, não correspondem a nenhuma criatura.

Ajude os *Fighters* a descobrir quantos *Foos* foram detectados, para que levem a quantidade exata de potes para transportá-los.

Entrada

A primeira linha da entrada contém um inteiro n ($1 \leq n \leq 1000$), indicando o número de caracteres que foram impressos pelo programa de identificação. A segunda linha da entrada contém uma string de letras minúsculas S ($|S| = n$), representando a saída do programa.

Saída

Determine o número de potes que devem ser usados pelos *Fighters* para capturar todos os *Foos*.

Exemplo

Entrada	Saída
18 fooscontrafighters	1
22 fofoofoofoomuitosfoos	4
20 abracadabrapedecabra	0

Notas

No primeiro exemplo, é necessário somente um pote: fooscontrafighters

No segundo exemplo, são necessários quatro potes: fofoofoofoomuitosfoos

No terceiro exemplo, não é necessário nenhum pote.

Problema G – Guerra nas Estrelas

Autor: Arthur Andrade D’Olival

Em uma galáxia muito, muito distante, um jovem aprendiz está prestes a iniciar sua jornada para se tornar um Jedi. Antes de partir, ele recebe uma palavra relacionada à saga. Independentemente da palavra recebida, o aprendiz deve responder com a mesma mensagem de incentivo: **Que a Força esteja com voce!**.

Sua tarefa é escrever um programa que imprima a mensagem **Que a Força esteja com voce!**.



Figura 1: Baby Yoda.

Entrada

A entrada contém uma única linha com uma palavra. A palavra possui de 1 a 100 letras, maiúsculas ou minúsculas, sem espaços.

Saída

Imprima uma única linha contendo exatamente a frase abaixo, sem acentos:

Que a Força esteja com voce!

Exemplo

Entrada	Saída
Jedi	Que a Força esteja com voce!

Notas

No exemplo, recebemos a palavra **Jedi** e respondemos com **Que a Força esteja com voce!**, independentemente da entrada.

Problema H – Hertz pra todos

Limite de tempo: 2s
Limite de memória: 256MB

Autor: Igor de Oliveira Moura

Na década de 1940, quando muitas residências já possuíam aparelhos de rádio, o engenheiro Arthur Lauro Xavier Moura percebeu um problema: mesmo tendo um rádio, algumas famílias não conseguiam utilizá-lo por não receberem sinal de transmissão.

Para resolver esse problema, o engenheiro criou o projeto *Hertz pra Todos*, cujo objetivo era ampliar a cobertura de rádio e permitir que mais residências recebessem sinal.

Para organizar o projeto, ele representou a região em um plano cartesiano, contendo n residências e m torres de transmissão já instaladas.

Cada residência está localizada em um ponto (x_i, y_i) . Cada torre de transmissão possui uma posição (a_i, b_i) e um raio de cobertura r_i .

Uma residência recebe sinal de uma torre se ela estiver dentro da sua área de alcance.

Para ampliar a cobertura, o engenheiro decidiu instalar uma nova torre, denominada **Torre X**, cujo raio de cobertura é R .

A Torre X pode ser instalada em qualquer ponto (x, y) com coordenadas inteiras, satisfazendo $0 \leq x, y \leq 1000$.

Sua tarefa é determinar a posição da Torre X que **maximize a quantidade total de residências atendidas**, considerando tanto as torres existentes quanto a nova torre.

Caso existam várias posições que proporcionem a mesma quantidade máxima de residências atendidas, escolha a posição com o menor valor de x . Em caso de empate, escolha aquela com o menor valor de y .

Se todas as residências já estiverem recebendo sinal das torres existentes, não será necessário instalar a Torre X.

Entrada

A primeira linha contém três inteiros n , m e R ($1 \leq n \leq 10^4$, $1 \leq m \leq 10^3$, $1 \leq R \leq 1000$), representando, respectivamente, a quantidade de residências, a quantidade de torres existentes e o raio de cobertura da Torre X.

As próximas n linhas contêm dois inteiros x_i e y_i ($0 \leq x_i, y_i \leq 1000$), representando as coordenadas de cada residência.

As próximas m linhas contêm três inteiros a_i , b_i e r_i ($0 \leq a_i, b_i \leq 1000$, $1 \leq r_i \leq 1000$), representando as coordenadas e o raio de cobertura de cada torre existente.

Saída

Caso todas as residências já possuam sinal, imprima um único inteiro -1 .

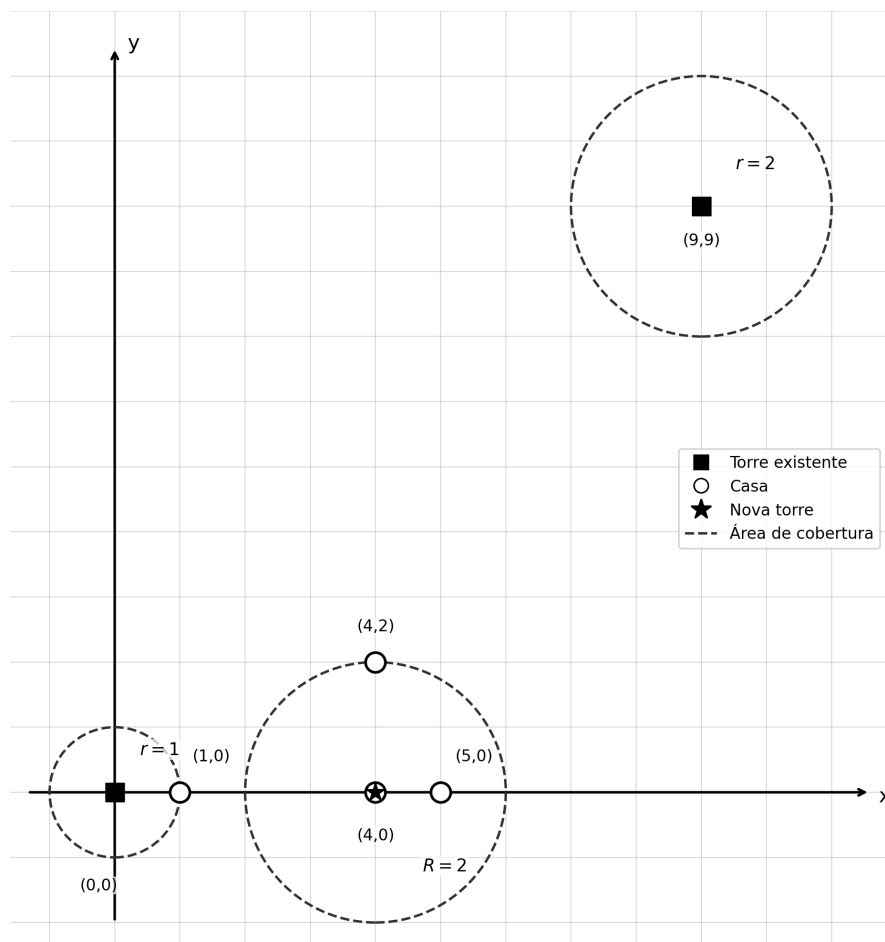
Caso contrário, imprima três inteiros q , x e y , em que q representa a quantidade máxima de residências atendidas após a instalação da Torre X, e (x, y) representa a posição escolhida.

Exemplo

Entrada	Saída
6 2 2	6 4 0
0 0	
1 0	
4 0	
5 0	
4 2	
9 9	
0 0 1	
9 9 2	

Notas

As residências $(0,0)$, $(1,0)$ e $(9,9)$ já recebem sinal das torres existentes. Instalando a Torre X em $(4,0)$ com $R = 2$, as residências $(4,0)$, $(5,0)$ e $(4,2)$ ficam a distâncias 0, 1 e 2 da nova torre, e todas as 6 ficam atendidas. A posição $(4,1)$ também serviria, mas $(4,0)$ tem o menor y entre as que têm o menor x possível. A figura abaixo ilustra o exemplo dado:



Problema I – Indo a Uberlândia

Autor: Daniel Saad Nogueira Nunes

As melhores equipes da UnB e UnB/FCTE se classificaram para a final brasileira da 31^a Maratona SBC de programação, que será em Uberlândia. Para sair de Brasília e ir à Uberlândia, os times precisarão pegar uma estrada que passar por várias cidades. A cada x quilômetros percorridos, há um pedágio que custa R\$ 7.50, mas não há pedágio em Uberlândia. Calcule o valor total gasto em pedágio pelas equipes.

Entrada

A entrada possui dois inteiros x e y ($1 \leq x, y \leq 10^6$), representando, respectivamente, a distância entre os pedágios e a distância total no trajeto entre Brasília e Uberlândia.

Saída

Imprima o valor total gasto em pedágio pelas equipes no formato “R\$ X.XX”.

Exemplo

Entrada	Saída
30 100	R\$ 22.50
50 250	R\$ 30.00
700 550	R\$ 0.00

Problema J – Juntando os Dígitos

Autor: Gabriel César Silvino Xavier

Um estagiário no Banco do Brasil recebeu a tarefa de escrever um programa que gere dados para testar o cadastro de cartões de crédito.

Para montar um teste, o estagiário recebeu uma sequência com alguns dígitos já escolhidos e outros marcados com **x**.

O sistema usa o algoritmo de Luhn para verificar os números e detectar erros de digitação. Nessa verificação, os dígitos são contados da direita para a esquerda, e o segundo, o quarto, o sexto e assim por diante são dobrados. Se um valor dobrado ultrapassar 9, subtrai-se 9 dele. A sequência é válida quando a soma dos valores obtidos, incluindo os dígitos que não foram dobrados, é divisível por 10.

Ajude o estagiário a completar a sequência. Cada ocorrência de **x** deve ser substituída por um dígito de 1 a 9. É garantido que existe uma resposta, e qualquer sequência válida será aceita.

Entrada

A entrada contém uma sequência S ($2 \leq |S| \leq 2 \cdot 10^5$) em uma linha, com dígitos de 1 a 9 e pelo menos dois **x**.

Saída

Imprima uma sequência válida em uma linha. Substitua cada **x** por um dígito de 1 a 9 e mantenha os dígitos conhecidos.

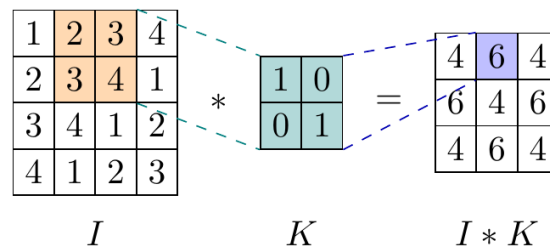
Exemplo

Entrada	Saída
7x42x8	714238
x472x	24729

Problema K – Kernel

Autor: Gabriel César Silvino Xavier

Yann está criando um editor de fotos e quer usar convolução para aplicar filtros às imagens. Para isso, ele percorre a imagem com uma matriz menor, chamada *kernel*, e soma os produtos dos valores sobrepostos em cada posição em que ela cabe inteira na imagem.



Ajude Yann a implementar a convolução no seu editor de fotos.

Entrada

A primeira linha contém os inteiros H e W ($1 \leq H, W \leq 1024$), representando, respectivamente, o número de linhas e colunas da imagem. As próximas H linhas contêm a imagem, uma matriz de inteiros $H \times W$ com valores entre 0 e 255.

Em seguida, um inteiro K ($1 \leq K \leq \min(H, W, 10)$) indica o tamanho do kernel. As próximas K linhas contêm o kernel, uma matriz de inteiros $K \times K$ com valores entre -10 e 10 .

Saída

Imprima a matriz resultante, com $H - K + 1$ linhas e $W - K + 1$ colunas, separando os valores de cada linha por espaços.

Exemplo

Entrada	Saída
3 3	2 4 6
1 2 3	4 6 8
2 3 4	6 8 2
3 4 1	
1	
2	
3 3	6 8
2 3 4	8 2
3 4 1	
4 1 2	
2	
0 1	
1 0	
4 4	4 3
2 2 2 2	6 5
2 1 2 2	
2 3 4 2	
2 2 2 2	
3	
0 0 0	
1 0 1	
0 0 0	

Problema L – Leitura de Relatórios

Autor: Daniel Saad Nogueira Nunes

A empresa Blackstar trabalha na área de análise de dados e possui um sistema que gera relatórios financeiros de seus clientes. Cada relatório é composto por uma sequência de números inteiros, representando os valores de transações financeiras.

O jovem Lucas percebeu que o sistema da empresa está gerando relatórios com zeros à esquerda, o que não é desejável. Ele decidiu escrever um programa para remover esses zeros à esquerda dos números nos relatórios. Especialmente, se o número for composto apenas por zeros, ele deve ser representado como um único zero.

Ajude Lucas a escrever esse programa.

Entrada

A primeira linha da entrada possui um inteiro n ($1 \leq n \leq 10^5$), representando a quantidade de dígitos do relatório. A segunda linha da entrada possui uma sequência de n dígitos, representando o relatório financeiro.

Saída

Imprima o relatório financeiro sem os zeros à esquerda. Se o relatório for composto apenas por zeros, imprima um único zero.

Exemplo

Entrada	Saída
5 12345	12345
4 0000	0
5 00123	123

Problema M – Mentalista

Autor: Arthur Andrade D'Oliveira

No **Instituto Federal de Brasília**, o troféu da **Maratona Padawan de Programação** desapareceu pouco antes da premiação. No lugar dele, a organização encontrou apenas um bilhete misterioso, o suficiente para **Patrick Jane** se interessar pelo caso e para **Lisbon** começar a se arrepender de tê-lo deixado chegar perto da investigação.

Uma câmera registrou o **horário exato do roubo**, mas, para variar, a gravação não ajudava a identificar o responsável. Felizmente, a organização possuía os horários de entrada e saída de todas as pessoas que passaram pelo local naquele dia.

Antes de recorrer a truques psicológicos, provocações ou alguma ideia que certamente irritaria Lisbon, Jane decidiu reduzir a lista de suspeitos: quem não estava no local no instante do roubo poderia ser descartado.

Sua tarefa é listar todas as pessoas que estavam presentes no horário do roubo.

Uma pessoa é considerada **suspeita** se o horário do roubo estiver entre seus horários de entrada e saída, **incluindo os dois extremos**. Estar na lista não significa ser culpado, significa apenas que Jane ainda não decidiu qual truque psicológico usar para descobrir a verdade.



Figura 1: Patrick Jane em miniatura.

Entrada

A primeira linha contém o horário T em que o troféu foi roubado.

A segunda linha contém um inteiro N ($1 \leq N \leq 100$), o número de pessoas com registros de presença.

Cada uma das próximas N linhas contém um nome e dois horários, separados por espaços, indicando quando a pessoa entrou e saiu do local, respectivamente.

Cada nome tem entre 1 e 20 letras maiúsculas ou minúsculas do alfabeto, sem espaços ou acentos. Todos os nomes são distintos; letras maiúsculas e minúsculas são diferenciadas.

Todos os horários são dados no formato HH:MM, com dois dígitos para horas e minutos, entre 00:00 e 23:59. Todos pertencem ao mesmo dia. Para cada pessoa, é garantido que o horário de entrada é menor ou igual ao de saída.

Saída

Imprima os nomes de todas as pessoas suspeitas, um por linha, em **qualquer ordem**. Cada nome deve aparecer exatamente uma vez, com a mesma grafia usada na entrada.

Se nenhuma pessoa estava no local no horário do roubo, imprima uma linha com “-1”.

Exemplo

Entrada	Saída
14:20	Clara
5	Alice
Clara 14:00 14:25	Elisa
Bruno 13:00 14:19	
Alice 14:10 15:00	
Diego 14:41 16:00	
Elisa 14:00 15:00	
10:00	-1
2	
Rafael 09:00 09:59	
Lia 10:01 11:00	
08:00	Pedro
3	Ana
Pedro 07:30 08:00	Caio
Ana 08:00 08:30	
Caio 08:00 08:00	

Notas

No primeiro exemplo, o roubo ocorreu às 14:20. Clara, Alice e Elisa estavam no local nesse horário. Bruno já havia saído e Diego ainda não havia chegado. Os três nomes podem ser impressos em qualquer ordem.

No segundo exemplo, Rafael saiu antes das 10:00 e Lia chegou depois. Portanto, não há suspeitos.

No terceiro exemplo, Pedro saiu exatamente às 08:00, Ana entrou nesse horário e Caio entrou e saiu nesse mesmo instante. Como os extremos são incluídos, os três são suspeitos.