



2025
Caderno de Problemas
XII Maratona Mineira de Programação

Patrocínio



Realização



Organização



Informações gerais

Este caderno de tarefas é composto por 24 páginas (não contando a capa), numeradas de 1 a 24. Verifique se o caderno está completo.

Entrada

- A entrada deve ser lida da entrada padrão.
- A entrada consiste em exatamente um caso de teste, que é descrito usando uma quantidade de linhas que depende do problema. O formato da entrada é como descrito em cada problema. A entrada não contém nenhum conteúdo extra.
- Todas as linhas da entrada, incluindo a última, terminam com o caractere de fim de linha (\n).
- A entrada não contém linhas vazias.
- Quando a entrada contém múltiplos valores separados por espaços, existe exatamente um espaço em branco entre dois valores consecutivos na mesma linha.

Saída

- A saída deve ser escrita na saída padrão.
- A saída deve respeitar o formato especificado no enunciado. A saída não deve conter nenhum dado extra.
- Todas as linhas da saída, incluindo a última, devem terminar com o caractere de fim de linha (\n).
- Quando uma linha da saída apresentar múltiplos valores separados por espaços, deve haver exatamente um espaço em branco entre dois valores consecutivos.
- Quando um valor da saída for um número real, use pelo menos o número de casas decimais correspondente à precisão requisitada no enunciado.

Problemas Interativos

A prova pode conter problemas interativos. Nesse tipo de problema, os dados de entrada fornecidos ao seu programa podem não ser predeterminados, mas são construídos especificamente para a sua solução. O juiz escreve um programa especial (o interador), cuja saída é transferida para a entrada da sua solução, e a saída do seu programa é enviada para a entrada do interador. Em outras palavras, sua solução e o interador trocam dados e podem decidir o que imprimir com base no histórico da comunicação.

Quando você escreve a solução para um problema interativo, é importante lembrar que, se você imprimir algum dado, é possível que ele seja primeiro armazenado em um *buffer* interno e não seja transferido imediatamente para o interador. Para evitar essa situação, você deve usar uma operação especial de *flush* toda vez que imprimir algum dado. Essas operações de *flush* estão presentes nas bibliotecas padrão de quase todas as linguagens:

- `fflush(stdout)` em C.
- `cout.flush()` em C++.
- `sys.stdout.flush()` em Python.
- `System.out.flush()` em Java.

Problema A. Antissocial

Hoje é dia de show! Os fãs estão ansiosos para ouvir seus músicos favoritos: Clube da Esquina.



Apesar de apaixonados pela música, os N fãs que chegarão um por vez no show não gostam de contato entre si (são definitivamente antissociais). Devido a isso, decidiram ocupar cada uma das N cadeiras usando uma regra simples: “escolher a posição que maximiza a distância para o fã mais próximo já sentado e, em caso de empate, escolher a cadeira de menor índice” (exceto para o primeiro espectador, que sempre ocupará a cadeira de número 1). Note que, para este show, as cadeiras foram dispostas em forma de uma única fileira.

Dada a quantidade N de fãs (e cadeiras), sua tarefa é determinar a ordem final dos fãs.

Entrada

A entrada é composta apenas por um número inteiro N ($1 \leq N \leq 3 \times 10^5$), a quantidade de fãs.

Saída

A saída deve conter uma linha composta de N números inteiros, em que o i -ésimo valor da linha indica qual fã (com base na ordem de chegada) ocupa a i -ésima cadeira.

Exemplos

Exemplo de entrada 1

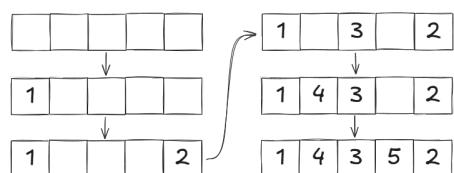
5

Exemplo de saída 1

1 4 3 5 2

Explicação do exemplo 1

A primeira pessoa a chegar ocupará a cadeira 1. A segunda pessoa a chegar escolherá a cadeira 5, pois é a mais distante da cadeira 1. A terceira pessoa escolherá a cadeira 3, já que esta está a uma distância 2 das cadeiras 1 e 5. A quarta pessoa escolherá a cadeira 2, uma vez que está a uma distância 1 das cadeiras 1 e 3 (a cadeira 4 tem a mesma menor distância, mas escolhemos a de menor índice). A última pessoa ocupará a cadeira 4. Desta forma, obtemos a ordem [1, 4, 3, 5, 2].



Exemplo de entrada 2

8

Exemplo de saída 2

1 5 6 3 7 4 8 2

Problema B. Bares



Luffy é um entusiasta de suco de laranja, porém ele já gastou toda a sua mesada e não consegue comprar nenhum copo. Para satisfazer sua paixão, mesmo sem dinheiro, ele foi para a Rua Sapucaí.

A Rua Sapucaí é famosa por sua grande quantidade de bares, sendo composta por uma sequência deles de maneira que Luffy consegue ir do i -ésimo bar ao j -ésimo bar em $|i - j|$ segundos. Os clientes desses bares têm a tradição de comemorar boas notícias comprando uma rodada de suco de laranja para todos que estão no mesmo bar naquele segundo. Luffy conhece muito bem os clientes, então sabe exatamente em qual bar x_i e em qual segundo t_i cada rodada de suco vai ocorrer.

Ajude-o a calcular o número máximo de sucos de laranja que ele pode beber, se ele escolher onde ele vai começar e andar de maneira ótima. Luffy não precisa parar no bar para beber o suco, pois sabe andar e beber ao mesmo tempo.

Entrada

A primeira linha da entrada contém um inteiro N ($1 \leq N \leq 10^6$), o número de rodadas de suco. As próximas N linhas contêm pares de inteiros: na i -ésima delas, x_i ($0 \leq x_i \leq 10^6$) e t_i ($0 \leq t_i \leq 10^6$), indicando que uma rodada de suco ocorrerá no bar x_i no segundo t_i . É garantido que não ocorrem duas rodadas em um mesmo bar no mesmo segundo.

Saída

A saída deve conter um único inteiro, o número máximo de sucos de laranja que Luffy pode beber.

Exemplos

Exemplo de entrada 1

3	
1 0	
0 1	
1 2	

Exemplo de saída 1

3	
---	--

Exemplo de entrada 2

4	
0 1	
1 0	
2 2	
3 3	

Exemplo de saída 2

3	
---	--

Explicação do exemplo 2

Luffy consegue participar de até 3 rodadas na ordem: $(1, 0), (2, 2), (3, 3)$.

Problema C. Cartas

Truco, ladrão!



O Truco Mineiro é o mais querido jogo de cartas do estado, entretanto, ele é notoriamente difícil de se jogar por conta de suas regras esquisitas. Truco é jogado com apenas um baralho (nenhuma carta se repete mas algumas não são usadas no jogo) contendo os tipos de 2 a 7, os ás, as damas, os valetes e os reis (A, Q, J, K, respectivamente). Todos os naipes de todas as cartas anteriores são incluídos (copas - C, espadas - E, ouros - O e paus - P). Uma carta é representada pela concatenação do seu tipo e seu naipe (4P é o quatro de paus).

A ordem de força das cartas no jogo é a seguinte (da mais forte para a mais fraca):

4P	7C	AE	70	3	2	A	K	J	Q	7	6	5	4
----	----	----	----	---	---	---	---	---	---	---	---	---	---

Na tabela, nas posições em que o naipe foi omitido, todos os naipes daquele tipo (exceto os citados anteriormente) têm a mesma força (como exemplo, 4P é a carta mais forte e 4C, 4E e 4P estão empatadas como as cartas mais fracas).

Para ensinar seus amigos a jogar, Riverson propôs que os amigos jogassem uma rodada simplificada do jogo seguindo suas dicas. Depois de aprenderem o básico, os amigos de Riverson vão poder jogar o Truco oficial, fazendo uso de mentiras, gritos, estratégias, tapas na mesa e discórdia para ganhar!

A versão simplificada de Riverson é a seguinte:

- É um jogo de duplas jogado com 4 jogadores (A, B, C, D), sendo as duplas A/C e B/D .
- Cada jogador começa com 3 cartas (é garantido que não terão cartas repetidas).
- A ordem dos jogadores em cada turno será $A \rightarrow B \rightarrow C \rightarrow D$ (sempre começando com A).
- A cada turno, cada jogador joga uma carta (o jogo dura 3 turnos).
- Ganha o turno a dupla que tiver jogado a carta mais forte (se houver empate entre as cartas mais fortes jogadas por duplas distintas, o turno termina em empate).

Dadas as regras do jogo, Riverson disse para cada jogador jogar da seguinte forma:

- Se a dupla do jogador estiver ganhando o turno ou se não for possível ganhar das cartas jogadas (não existe carta na mão do jogador mais forte que a carta mais forte jogada pelos oponentes): jogue sua pior carta.
- Caso contrário: jogue sua carta mais forte (quando não houver nenhuma carta na mesa, também deve-se jogar a mais forte).

Você deve imprimir, para cada turno, qual foi a dupla ganhadora ou se houve empate.

Entrada

A entrada é composta por 4 linhas representando as mãos de cada jogador (A, B, C, D , respectivamente). Cada linha tem por 3 sequências de dois caracteres cada, simbolizando a mão do jogador.

Saída

A saída deve conter três números, o resultado de cada um dos turnos: 1 se a dupla *A/C* tiver ganhado, 2 se a dupla *B/D* tiver ganhado e 0 caso o turno acabe em empate.

Exemplos

Exemplo de entrada 1

Exemplo de entrada 1	Exemplo de saída 1
70 7E AP QP 2C AE 7C JE 3C KO 4P 50	2 1 2

Explicação do exemplo 1

Turno 1: *A* - 70 / *B* - AE / *C* - 7C / *D* - 4P* (dupla *B/D* ganha)

Turno 2: *A* - A / *B* - 2 / *C* - 3* / *D* - 5 (dupla *A/C* ganha)

Turno 3: *A* - 7 / *B* - Q / *C* - J / *D* - K* (dupla *B/D* ganha)

Exemplo de entrada 2

Exemplo de entrada 2	Exemplo de saída 2
JC KP KO AC 60 5P 2P KC 3E 3P 30 3C	0 2 2

Explicação do exemplo 2

Turno 1: *A* - K / *B* - A / *C* - 3* / *D* - 3* (empate)

Turno 2: *A* - K / *B* - 5 / *C* - 2 / *D* - 3* (dupla *C/D* ganha)

Turno 3: *A* - J / *B* - 6 / *C* - K / *D* - 3* (dupla *C/D* ganha)

Exemplo de entrada 3

Exemplo de entrada 3	Exemplo de saída 3
AE 3C QP 3P QE 7E JC 3E JO 6E 7C 2P	2 1 0

Explicação do exemplo 3

Turno 1: *A* - AE / *B* - 7 / *C* - J / *D* - 7C* (dupla *B/D* ganha)

Turno 2: *A* - 3* / *B* - Q / *C* - J / *D* - 6 (dupla *A/C* ganha)

Turno 3: *A* - Q / *B* - 3* / *C* - 3* / *D* - 2 (empate)

Problema D. Dados



Eis que você e seus amigos acabam de pedir uma porção de torresmo mineiro em um boteco. Para ser justo, a quantidade de torresmo que cada um vai comer vai ser decidida pela sorte!

Um de seus amigos, por acaso, tinha 3 dados de 6 lados (com valores de 1 a 6) no bolso. Assim, para decidir quantos pedaços de torresmo cada um vai comer, serão jogados os 3 dados, e o valor sorteado será a soma dos valores dos dados.

Você então se pergunta: qual é a probabilidade da soma dos 3 dados ser igual a K ?

Entrada

A entrada é composta apenas por um inteiro K ($3 \leq K \leq 18$).

Saída

A saída deve conter apenas um número decimal: a probabilidade da soma dos 3 dados ser igual a K . A saída será considerada correta se estiver com um erro absoluto ou relativo menor que 10^{-3} em relação à saída do juiz.

Exemplos

Exemplo de entrada 1	Exemplo de saída 1
3	0.0046296296

Explicação do exemplo 1

Neste caso, há apenas uma forma em que a soma é 3: se todos os dados forem 1. Dessa forma, a probabilidade é $\frac{1}{6 \cdot 6 \cdot 6} \approx 0.00463$.

Exemplo de entrada 2	Exemplo de saída 2
10	0.125

Observações

Para especificar o número de casas decimais na saída, você pode usar:

- `printf("%.12lf\n", val)` em C.
- `cout << fixed << setprecision(12) << val` em C++.
- `print("%.12f" % val)` em Python.
- `System.out.printf("%.12f", val)` em Java.

Problema E. Emparelhamento

Qual doce vencerá este ano?



O Ultimate Battle de Açúcares (UBA) é um torneio realizado entre regiões de Minas Gerais para a disputa do melhor doce. Neste ano, o doce escolhido é o “Queijo com Goiabada” (também conhecido como Romeu e Julieta). Cada doce é composto por duas peças emparelhadas: uma de queijo e uma de goiabada. Cada uma das N regiões trouxe **exatamente um** doce, identificado por sua região.

Sabendo disso, um grupo de integrantes de uma associação anônima decidiu sabotar os doces, impedindo a realização do torneio. Para isso, eles trocaram peças (de queijo ou goiabada) entre pares de doces, deixando-os completamente misturados.

Para evitar esta tragédia, você foi convocado a ajudar na tarefa de emparelhar os pares de queijo com goiabada da mesma região antes que o torneio iniciasse. Para emparelhar os doces, podem ser feitas diversas operações, sendo cada operação composta de uma troca de uma peça (queijo ou goiabada) de um doce com outra peça (queijo ou goiabada) de outro doce.

Dadas as peças presentes em cada doce após a sabotagem, determine o menor número de operações necessárias para remontar os doces. **Note que não é necessário deixar os doces em ordem**, mas apenas remontá-los da forma correta conforme suas regiões.

Entrada

A primeira linha da entrada contém um inteiro N ($1 \leq N \leq 2 \times 10^5$) que representa o número de doces. Em cada uma das próximas N linhas são dados inteiros A e B ($1 \leq A, B \leq N$) que representam, respectivamente, as regiões de origem de cada uma das peças presentes no doce.

É garantido que cada número inteiro de 1 a N aparece exatamente duas vezes na descrição dos doces após a sabotagem.

Saída

A saída deve conter uma linha com um único inteiro, o menor número de operações necessárias para remontar os doces corretamente.

Exemplos

Exemplo de entrada 1

```
4
1 2
3 4
1 2
3 4
```

Exemplo de saída 1

```
2
```

Explicação do exemplo 1

Neste exemplo, podemos trocar a primeira peça do doce 1 com a segunda peça do doce 3, fazendo com que ambos os doces sejam formados por peças da mesma região. Da mesma forma, trocamos a segunda peça do doce 2 com a primeira peça do doce 4, também fazendo com que ambos os doces sejam formados por peças da mesma região. Portanto, são necessárias (no mínimo) 2 operações.

Exemplo de entrada 2

4
2 2
1 1
3 3
4 4

Exemplo de saída 2

0

Explicação do exemplo 2

Todos os doces já estão formados por peças da mesma região, portanto não é necessário fazer trocas.

Problema F. Fome de Queijo



Cientistas de uma universidade mineira estão estudando o comportamento de um rato quando exposto a um labirinto com alguns pedaços de queijo minas, com o interesse de avaliar a capacidade do rato de encontrar o queijo mais próximo.

O labirinto pode ser representado por uma grade de dimensões $N \times M$. A posição do rato está marcada pelo caractere R. Em algumas posições, representadas pelo caractere #, foram colocadas barreiras que impedem a passagem do rato, e outras contêm queijo minas, representadas pelo caractere Q. Todas as outras posições (representadas pelo caractere .) estão vazias.

Os cientistas posicionarão outros K pedaços de queijo no labirinto, mas ainda não decidiram em quais posições esses pedaços ficarão. O rato, que é muito esperto e tem muita fome, já está interessado em descobrir quantos movimentos serão necessários para chegar no queijo mais próximo. Em um movimento, o rato pode se mover para cima, para baixo, para a esquerda ou para a direita. O rato não pode passar por barreiras.

Descubra o valor esperado da distância do rato até o queijo mais próximo, considerando que a escolha do local dos K pedaços de queijo é feita de maneira uniformemente aleatória entre todas as escolhas possíveis.

Entrada

A primeira linha da entrada contém os inteiros N , M e K ($2 \leq N, M \leq 1000$ e $1 \leq K \leq NM - 1$).

As N linhas seguintes contêm M caracteres cada, representando a grade. Cada linha contém apenas os caracteres R, Q, . e #.

Em todos os casos, haverá exatamente uma posição com o caractere R e o rato será capaz de alcançar todas as posições vazias (e com queijo) da grade através de um ou mais movimentos. Além disso, haverá pelo menos K posições vazias.

Saída

A saída deve conter somente um único inteiro: a distância esperada do rato até o queijo mais próximo, módulo 998244353.

Formalmente, seja $M = 998244353$. É possível provar que a resposta exata pode ser expressa como uma fração irredutível $\frac{p}{q}$, em que p e q são inteiros e $q \not\equiv 0 \pmod{M}$. Imprima o inteiro igual a $p \cdot q^{-1} \pmod{M}$. Ou seja, imprima o inteiro x tal que $0 \leq x < M$ e $x \cdot q \equiv p \pmod{M}$.

Exemplos

Exemplo de entrada 1

```
3 3 1
R..
##.
Q..
```

Exemplo de saída 1

```
3
```

Explicação do exemplo 1

Neste caso, temos 5 células livres, e um queijo será adicionado. A distância até o queijo mais próximo será, então, 1, 2, 3, 4 ou 5, cada valor com probabilidade $\frac{1}{5}$. Dessa forma, a distância esperada é $\frac{1}{5} \times (1 + 2 + 3 + 4 + 5) = \frac{1}{5} \times 15 = 3$.

Exemplo de entrada 2

```
2 2 2
R.
..
```

Exemplo de saída 2

```
1
```

Explicação do exemplo 2

Neste caso, 2 queijos serão adicionados. Independente de qual forma, podemos notar que o rato sempre estará a distância 1 de um queijo, portanto a resposta é 1.

Exemplo de entrada 3

```
4 4 1
##.#
#. .R
Q#.#
...Q
```

Exemplo de saída 3

```
427819011
```

Explicação do exemplo 3

Neste caso, o valor esperado é $\frac{18}{7}$. A resposta é 427819011, uma vez que $427819011 \times 7 \equiv 18 \pmod{998244353}$.

Problema G. GPS

Nú, vei, não acredito que o motorista vai ter que dar essa volta toda. Eu poderia só andar um quarteirão que seria muito mais rápido!



Você se encontra em uma cidade, que pode ser representada por um conjunto de N esquinas (vértices) conectados por dois tipos de aresta: C ruas de mão única percorridas por um carro em tempo c_i e P calçadas de pedestre percorridas a pé **em ambas as direções** em tempo p_i . Você está no vértice 1 e quer ir para o vértice N .

Você quer chamar um UBERaba (aplicativo de transporte usado em Minas Gerais) para chegar ao seu destino. Porém, para chegar mais rápido, você está disposto a caminhar no início e no fim da viagem, da seguinte forma: você está disposto a caminhar até alguma esquina x (pode ser que $x = 1$), depois o UBERaba vai te levar da esquina x até uma esquina y (pode ser que $x = y$), e por fim você vai caminhar para chegar no destino (pode ser que $y = N$). Como ele usa GPS, o UBERaba sempre vai tomar a rota mínima de x a y . Considere que você consegue sincronizar tudo perfeitamente, de forma que você pode encontrar o UBERaba na esquina x imediatamente.

Dentre todas as possibilidades de rota, calcule o tempo mínimo para sair do vértice 1 e chegar no vértice N , dadas as restrições.

Entrada

A primeira linha da entrada contém três inteiros N , C e P ($2 \leq N \leq 5 \times 10^5$, $0 \leq C, P \leq 5 \times 10^5$).

As próximas C linhas contêm 3 inteiros cada, a_i, b_i, c_i , indicando que existe uma rua de mão única da esquina a_i para a esquina b_i ($1 \leq a_i, b_i \leq N$, $a_i \neq b_i$) percorrida por um carro em tempo c_i ($1 \leq c_i \leq 10^9$).

As próximas P linhas contêm 3 inteiros cada, a_i, b_i, p_i , indicando que existe uma calçada da esquina a_i para a esquina b_i ($1 \leq a_i, b_i \leq N$, $a_i \neq b_i$) percorrida a pé em ambas as direções em tempo p_i ($1 \leq p_i \leq 10^9$).

É garantido que não existe aresta de um vértice para ele mesmo ou arestas repetidas. É garantido também que a entrada é tal que existe pelo menos uma forma de sair do vértice 1 e chegar no vértice N .

Saída

A saída deve conter um único inteiro: o menor tempo possível para sair do vértice 1 e chegar no vértice N .

Exemplos

Exemplo de entrada 1

5 6 4
1 3 5
2 3 1
3 4 1
3 5 7
4 3 8
4 5 5
2 1 2
2 4 5
3 5 7
4 5 2

Exemplo de saída 1

6

Explicação do exemplo 1

Neste caso, uma solução ótima é fazer o trajeto da seguinte forma:

- $1 \xrightarrow{2} 2$ a pé, em 2 minutos;
- $2 \xrightarrow{1} 3 \xrightarrow{1} 4$ de carro, em $1 + 1 = 2$ minutos;
- $4 \xrightarrow{2} 5$ a pé, em 2 minutos.

O trajeto completo é feito em $2 + 2 + 2 = 6$ minutos.

Exemplo de entrada 2

3 2 2
1 2 1
2 3 1
1 2 5
2 3 5

Exemplo de saída 2

2

Explicação do exemplo 2

Neste caso, o tempo mínimo é atingido ao se fazer o trajeto inteiro de carro, em $1 + 1 = 2$ minutos.

Exemplo de entrada 3

3 0 2
1 2 1
2 3 1

Exemplo de saída 3

2

Explicação do exemplo 3

Neste caso, a única possibilidade é fazer o trajeto inteiro a pé, em $1 + 1 = 2$ minutos.

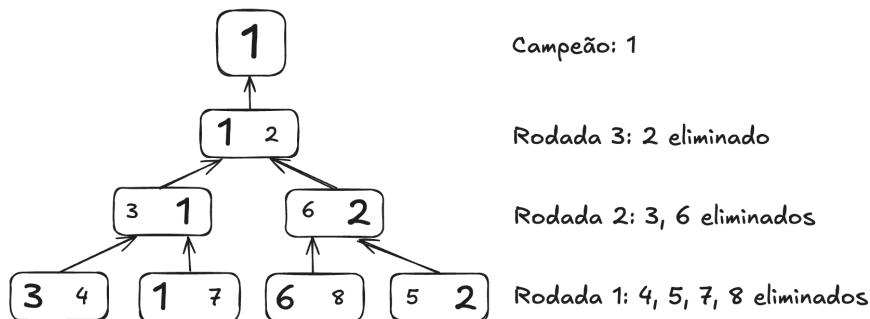
Problema H. Horóscopo



Tobias é o organizador do Torneio Mineiro de Totó, o Totórneo Mineiro. O Totórneo é composto por 2^N times (identificados por inteiros $1, \dots, 2^N$) e é jogado no formato “mata-mata” (logo, o campeonato tem N rodadas):

- Ao perder, o time é eliminado do campeonato.
- Cada rodada é composta por uma lista dos times restantes e, seguindo essa lista, o time na primeira posição joga com o na segunda posição, o na terceira posição joga com o na quarta posição e assim sucessivamente.
- A ordem relativa dos ganhadores em uma rodada é mantida para a rodada seguinte.

O ritmo acelerado e altamente técnico faz do totó um jogo extremamente determinístico, de forma tal que é sempre possível saber o vencedor da partida. Quando dois times $i, j \in \{1, \dots, 2^N\}$ competem, é garantido que i vence j se e somente se $i < j$. Segue o exemplo de um campeonato (note que, como o jogo é determinístico, o chaveamento inicial define os resultados):



João, o amigo astrólogo de Tobias, acabou de montar um serviço de Horóscopos por SMS e mandou para os assinantes sua previsão para o Totórneo. A previsão é surpreendentemente detalhada e diz, para cada time, em qual rodada r (um número de 1 a N) ele será eliminado ou 0 caso seja previsto que o time será campeão.

Para ajudar seu amigo, Tobias decidiu abrir mão de sua integridade profissional e vai montar o chaveamento inicial para que as previsões de João se concretizem. Sua tarefa é gerar um chaveamento compatível com as previsões ou detectar que é impossível.

Entrada

A primeira linha da entrada é composta pelo inteiro N ($1 \leq N \leq 18$), o número de rodadas do campeonato (consequentemente, o campeonato terá 2^N times).

A segunda linha é composta por 2^N inteiros r_i ($1 \leq r_i \leq N$) sendo r_i a rodada que o astrológo João previu em que o i -ésimo time sairia do campeonato ou 0 caso tenha previsto que o time será

ganhador. É garantido que o número de times eliminados será consistente com o proposto pelo campeonato (apenas um campeão, exatamente metade dos times eliminados na primeira rodada, um quarto dos times eliminados na segunda e assim sucessivamente).

Saída

Caso seja possível montar um chaveamento inicial tal que as previsões de João se concretizem, a saída deve conter o número dos times t_1, \dots, t_{2^N} em uma ordem do chaveamento compatível com as previsões de João. Se houver mais de uma ordem possível, qualquer uma será aceita.

Caso seja impossível montar tal chaveamento, imprima -1 .

Exemplos

Exemplo de entrada 1

2 0 1 2 1	1 2 3 4
--------------	---------

Exemplo de saída 1

Explicação do exemplo 1

Utilizando a ordem $[1, 2, 3, 4]$, temos dois embates na primeira rodada: o time 1 joga contra o time 2 (1 sai como vencedor) e o time 3 joga contra o time 4 (3 sai como vencedor). Na final temos o embate entre time 1 e time 3, com o time 1 saindo vencedor.

Desta forma, como previsto por João, temos que o time 1 foi o vencedor, os times 2 e 4 foram eliminados na primeira rodada e o time 3 foi eliminado na segunda rodada.

Exemplo de entrada 2

1 1 0	-1
----------	----

Exemplo de saída 2

Explicação do exemplo 2

Nesse caso é esperado que o time 1 seja eliminado na primeira rodada e que o time 2 seja o campeão, o que é impossível dado que o time 1 sempre vence o time 2.

Exemplo de entrada 3

3 0 3 2 1 1 2 1 1	3 4 1 7 6 8 5 2
----------------------	-----------------

Exemplo de saída 3

Problema I. Interativo

Uai, que trem diferente, sô!



Este problema é interativo.

O sistema de iluminação do local de prova da Maratona Mineira é muito tecnológico. Foram instaladas N^2 lâmpadas de LED, dispostas em uma grade de N linhas, com N colunas de lâmpadas cada. As lâmpadas são controladas por um sistema de controle remoto, que aceita dois tipos de instruções:

- A instrução $L i$ troca o estado de todas as lâmpadas da linha i , ou seja, se a lâmpada estava acesa, ela passa a estar apagada e vice-versa. Ao executar essa instrução, o sistema exibe na tela o número de lâmpadas acesas na linha i **após** a operação.
- A instrução $C j$ troca o estado de todas as lâmpadas da coluna j , ou seja, se a lâmpada estava acesa, ela passa a estar apagada e vice-versa. Ao executar essa instrução, o sistema exibe na tela o número de lâmpadas acesas na coluna j **após** a operação.

A equipe de iluminação precisa da sua ajuda! A prova está prestes a começar, e todas as lâmpadas do local de prova estavam acesas, mas alguém invadiu o sistema de controle e realizou uma série de instruções arbitrárias, deixando as lâmpadas em uma configuração desconhecida.

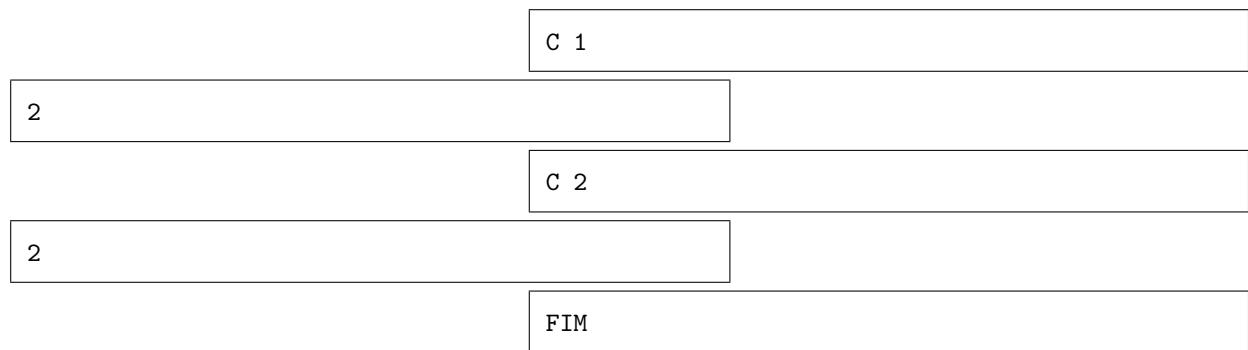
Faça todas as lâmpadas ficarem acesas usando no máximo $5N$ instruções. **É garantido que é possível acender todas as lâmpadas da grade a partir da configuração inicial.**

Interação

Inicialmente, está disponível para a leitura um único inteiro N ($1 \leq N \leq 100$). Após sua leitura, você pode fazer no máximo $5N$ instruções, que são da forma “ $L i$ ” (sem aspas) para trocar o estado da linha i ($1 \leq i \leq N$), ou “ $C j$ ” (sem aspas) para trocar o estado da coluna j ($1 \leq j \leq N$). Após cada pergunta, ficará disponível para a leitura o número de lâmpadas acesas na linha ou coluna especificada **após** a operação. Quando souber que todas as lâmpadas estão acesas, imprima FIM e termine o programa.

Exemplos

Leitura	Exemplo de interação 1	Escrita
2		
0	L 1	



Explicação do exemplo 1

Inicialmente só as lâmpadas da primeira linha estão acesas. Após as duas últimas instruções, podemos ter certeza de que todas as lâmpadas estão acesas.

Observações

O interador **não é adaptativo**, ou seja, o estado do problema não muda durante a interação.

Após uma escrita, não se esqueça de dar *flush* no *buffer* de saída. Caso contrário, você receberá o veredito TIME LIMIT EXCEEDED. Para fazer isso, use:

- `fflush(stdout)` em C.
- `cout.flush()` em C++.
- `sys.stdout.flush()` em Python.
- `System.out.flush()` em Java.

Problema J. Jeito de Falar

Sinto a fúria em suas palavras, mas não entendo nada do que você diz.

– William Shakespeare



Bira viajou para o interior de Minas, e agora precisa pegar um trem. Ao chegar na estação, ele pergunta ao funcionário qual é o número da estação, para verificar se está na estação certa.

Porém, os moradores locais têm um costume diferente: eles falam números da maneira codificada! Por exemplo, para falar o número 2020 eles poderiam dizer 220 (duas vezes o número 20), e o número 20555 poderia ser codificado como 12045 (uma vez o número 20, seguido de quatro cópias do número 5).

Formalmente, uma codificação válida de um número X consiste em um número N , cujos dígitos podem ser particionados em $2k$ partes ($k \geq 1$): $P_1V_1P_2V_2 \dots P_kV_k$, de maneira que X pode ser representado pela concatenação de P_1 cópias do número V_1 , seguida de P_2 cópias do número V_2 , e assim por diante. Isto é, algumas partes do número dito podem ser consideradas como período (número de vezes a repetir), e irão repetir alguns dígitos seguintes. **Um período P_i nunca pode começar com o dígito 0.**

Ao perguntar o número da estação, o funcionário falou o número codificado N . Bira vai decodificar o número, porém podem existir muitas formas de decodificar. O número 321, por exemplo, pode ser decodificado como 212121 ou 111111111111111111111111111111. Bira não lembra o número exato de sua estação, mas lembra que ele tinha M dígitos.

Ajude Bira escrevendo um programa que, dado o número N falado pelo funcionário, calcula quantas formas existem de decodificar o número para que tenha M dígitos. Duas formas são consideradas distintas se possuírem número diferente de partes, ou alguma das partes for diferente.

Entrada

A primeira linha contém o número N ($2 \leq |N| \leq 2000$), onde $|N|$ representa o número de dígitos de N . É garantido que o primeiro dígito de N não é 0.

A segunda linha contém o número M ($1 \leq M \leq 2000$).

Saída

Imprima o número de formas de decodificar N com M dígitos. Como esse valor pode ser muito grande, imprima o resto de sua divisão por 998244353.

Exemplos

Exemplo de entrada 1

23225	2
6	

Exemplo de saída 1

Explicação do exemplo 1

Neste exemplo, existem duas formas de decodificar 23225 com 6 dígitos: 323255 e 332525.

Exemplo de entrada 2

321	0
3	

Exemplo de saída 2

Exemplo de entrada 3

10	1
1	

Exemplo de saída 3**Exemplo de entrada 4**

2001	1
6	

Exemplo de saída 4

Explicação do exemplo 4

A única decodificação de 2001 é 001001. Note que o número decodificado pode conter zeros à esquerda.

Exemplo de entrada 5

111111	3
4	

Exemplo de saída 5

Explicação do exemplo 5

Neste exemplo, há 3 maneiras de decodificar, e todas resultam no número final 1111. Representando o período entre colchetes e a parte repetida entre parênteses, as decodificações são 1[1](11), [1](11)[1](11) e [1](111)1.

Exemplo de entrada 6

16956758163750637	41
114	

Exemplo de saída 6

Problema K. Kubitschek

Meta 27: O programa de implantação da indústria automobilística exigirá um esforço financeiro que se estima em 38 bilhões de cruzeiros, incluído, nesta cifra, o equivalente a 263 milhões de dólares.

– *Programa de Metas do Presidente Juscelino Kubitschek*



JK, em seu ambicioso plano de crescer a indústria automobilística do país decidiu destruir locomotivas antigas em um trilho para reduzir a competição com o transporte rodoviário.

No trilho em questão pode passar qualquer uma dentre N diferentes locomotivas, cada uma descrita por uma posição inicial p e uma velocidade v . A locomotiva está na posição p no tempo 0 e, a cada instante de tempo, ela se move v (para a direita no trilho se v for positivo e para a esquerda se v for negativo).

Como JK não sabe qual locomotiva vai passar, ele decidiu instalar dinamite em algumas posições dos trilhos. Cada dinamite tem uma posição no trilho e será ativada por JK em um tempo específico (JK não consegue apertar mais de um detonador no mesmo instante, logo cada dinamite deve ser estourada em um tempo distinto). Formalmente, as dinamites são um conjunto de pares (x_i, t_i) (com t_i distintos), de forma que a dinamite i explode qualquer locomotiva que estiver exatamente na posição x_i no instante t_i .

Retorne uma lista de dinamites instaladas por JK de forma a garantir que, independente da locomotiva que passar no trilho, ela será destruída. É garantido que essa lista sempre existe. **Não é necessário minimizar a quantidade de dinamites.**

Entrada

A primeira linha da entrada é composta por um inteiro N ($1 \leq N \leq 1000$), o número de locomotivas.

Seguem N linhas, cada uma composta por dois inteiros p_i e v_i ($0 \leq |p_i|, |v_i| \leq 1000$) a posição e velocidade da i -ésima locomotiva, respectivamente.

Saída

A primeira linha da saída deve conter um inteiro $1 \leq M \leq 2000$, o número de dinamites que se decidiu instalar. As M linhas seguintes devem conter dois inteiros x_i, t_i ($0 \leq |x_i| \leq 2 \times 10^6$, $0 \leq t_i \leq 2000$) em cada uma: a posição da dinamite e o momento em que ela será explodida, respectivamente. É obrigatório que todos os t_i da resposta sejam distintos.

Se houver mais de uma resposta possível, qualquer uma será aceita. Não é necessário minimizar M .

Exemplos

Exemplo de entrada 1

Exemplo de saída 1
2 1 -1 -1 1

Explicação do exemplo 1

Tanto a primeira quanto a segunda locomotiva estão na posição 0 no instante 1, logo explodir uma dinamite na posição 0 no instante 1 é suficiente para explodir as locomotivas.

Exemplo de entrada 2

Exemplo de saída 2
3 0 1 0 2 0 3 4 3 1 8 4 3 3 4 2

Explicação do exemplo 2

A primeira locomotiva é destruída no instante 3, a segunda locomotiva nos instantes 2 (ou 4) e a terceira locomotiva no instante 1.

Exemplo de entrada 3

Exemplo de saída 3
3 0 1 0 2 0 3 1 0 0

Explicação do exemplo 3

Outra solução viável para o caso anterior: aqui todos as locomotivas são destruídas no instante 0.

Problema L. Lucro



Um mercado é dito eficiente quando os produtos são corretamente precificados, de forma tal que não é possível lucrar fazendo trocas dentro do mercado. Para testar a Hipótese do Mercado Eficiente, você foi a uma casa de câmbio e olhou os valores para trocas de três moedas:

- A : a taxa de conversão euro → real (quantos reais um euro vale)
- B : a taxa de conversão real → yuan chinês (quantos yuans chineses um real vale)
- C : a taxa de conversão euro → yuan chinês (quantos yuans chineses um euro vale)

Todas as conversões diretas são simétricas, significando que a mesma troca pode ser feita no sentido oposto (se um euro pode ser trocado por A reais, A reais podem ser trocados por um euro).

Como exemplo, seguindo as definições acima, o mercado com taxas de conversão $A = 2$, $B = 2$ e $C = 2$ não seria eficiente pois se pode fazer as conversões: 1 euro → 2 reais → 4 yuans chineses → 2 euros, conseguindo lucro somente convertendo as moedas.

Como um funcionário da Bovmesb (Bolsa de Valores Minas - Espírito Santo - Brasília), você deve responder se o mercado é eficiente, dadas as taxas de conversão.

Entrada

A entrada é composta por uma única linha contendo três números inteiros A, B, C ($1 \leq A, B, C \leq 10^4$) as taxas de conversão como descritas no enunciado.

Saída

Imprima S se o mercado for eficiente e N se o mercado não for eficiente.

Exemplos

Exemplo de entrada 1

2 3 6	
-------	--

Exemplo de saída 1

S

Exemplo de entrada 2

1 2 4	
-------	--

Exemplo de saída 2

N

Exemplo de entrada 3

2 2 2	
-------	--

Exemplo de saída 3

N

Problema M. Mar de Morros

Tô passando mal, tô passando mal!



Minas Gerais é um estado de muitos morros (ou, como muitos falam, um mar de morros). Fernanda deseja fazer uma viagem por Minas, porém ela sempre tem problemas com altitude e pode passar mal com a falta de oxigênio. O estado mineiro pode ser descrito como N cidades, conectadas por M estradas que podem ser percorridas em ambas as direções. Além disso, a cidade i tem uma altitude de h_i metros. Fernanda vai sair da cidade 1 e deseja chegar na cidade N .

A aclimatação acontece da seguinte forma: o corpo se adapta à altitude quando se dorme em uma cidade. Mais especificamente, ao se dormir em uma cidade i com altitude h_i , no dia seguinte Fernanda só pode visitar cidades com altitude em $[h_i, h_i + H]$, em que H é um valor fixo (dessa forma, ela sabe que não vai passar mal). Note que ela pode visitar várias cidades no mesmo dia.

Dada essa restrição, calcule o número mínimo de dias que Fernanda precisa para chegar na cidade N , ou imprima -1 se isso não é possível.

Entrada

A primeira linha da entrada contém três inteiros N, M, H ($2 \leq N \leq 10^5, 1 \leq M \leq 10^5, 1 \leq H \leq 10^9$). A segunda linha da entrada contém N inteiros: o i -ésimo deles representa o valor da altitude h_i ($1 \leq h_i \leq 10^9$). As próximas M linhas representam as estradas. A i -ésima delas contém dois inteiros a_i, b_i ($1 \leq a_i, b_i \leq N, a_i \neq b_i$), representando uma estrada conectando as cidades a_i e b_i . É garantido que não há estradas que conectam o mesmo par de cidades duas vezes na entrada, e também não há estradas que conectam uma cidade a ela mesma.

Saída

A saída deve conter um único inteiro: o número mínimo de dias necessários para sair do vértice 1 e chegar no vértice N , ou -1 se isso não é possível.

Exemplos

Exemplo de entrada 1

8 10 4
1 6 6 7 3 3 5 8
1 4
1 6
2 5
6 4
4 2
4 5
7 5
1 8
5 8
7 8

Exemplo de saída 1

3

Explicação do exemplo 1

Neste caso, Fernanda pode fazer os seguintes passos em cada dia:

1. $1 \rightarrow 6$;
2. $6 \rightarrow 4 \rightarrow 5 \rightarrow 7$;
3. $7 \rightarrow 8$.

Exemplo de entrada 2

2 1 1
1 1
1 2

Exemplo de saída 2

1

Explicação do exemplo 2

Neste caso, Fernanda pode ir direto no primeiro dia.

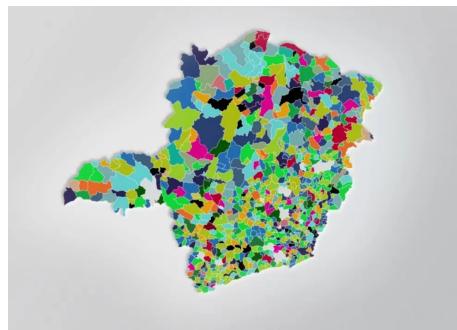
Exemplo de entrada 3

3 2 3
1 2 8
1 2
2 3

Exemplo de saída 3

-1

Problema N. Nome Mínimo



Você é prefeito de um município mineiro, cujo nome é S . Sua primeira ação após ser eleito é possivelmente modificar o nome do município para que ele fique mais alto na lista de municípios de Minas, que sempre é dada em ordem alfabética (lexicográfica), sem modificá-lo demais.

Dessa forma, você quer fazer **no máximo uma operação** de escolher um intervalo do nome e reverter esse intervalo. Por exemplo, revertendo o intervalo $[2, 3]$ de **abaete**, que representa **ba**, obtemos **aabete**.

Dado o nome S , calcule o menor nome possível em ordem lexicográfica, após realizar no máximo uma operação de reverter um intervalo de S .

Entrada

A entrada consiste em apenas uma linha, contendo o nome S do município ($1 \leq |S| \leq 10^6$). O nome consiste apenas de letras minúsculas de **a** a **z**.

Saída

A saída consiste em uma única linha com o lexicograficamente menor nome possível.

Exemplos

Exemplo de entrada 1

abaete	aabete
--------	--------

Exemplo de saída 1

Exemplo de entrada 2

uberaba	abarebu
---------	---------

Exemplo de saída 2

Explicação do exemplo 2

Neste caso, o menor nome é obtido ao reverter o nome inteiro.

Exemplo de entrada 3

luz	luz
-----	-----

Exemplo de saída 3

Explicação do exemplo 3

Neste caso, S já é o menor nome possível.