Problema A. ?Adreuqse arap atierid

Arquivo de entrada:	`entrada padrão`
Arquivo de saída:	`saída padrão`
Tempo limite:	1 segundo
Limite de memória:	256 megabytes

A codificação de caracteres Útil desde Cuneiforme até Siríaco (UCS) ou Unicode é uma invenção fenomenal que atualmente é desenvolvida em conjunto com o Instituto de Organização de Símbolos (ISO). O objetivo é possibilitar que todo tipo de símbolo e sistema de escrita que é de uso comum em todo o mundo hoje ou no passado seja representável nos computadores.

Esse esforço tem alguns desafios em seu caminho, sendo que um deles é que existem linguagens como o Árabe, o Hebreu e o Siríaco que escrevem da direita para a esquerda, sendo que os nossos computadores são habituados a entender as coisas da esquerda pra direita. O Consórcio do Unicode, para resolver tais problemas, criou alguns símbolos especiais e alguns algoritmos para que computadores entendam textos bidirecionais. Tome por exemplo este texto que mistura português e hebreu:

A ordem em que os caracteres são guardados é totalmente diferente da ordem em que eles são renderizados. Observe a ordem em que esses caracteres ficam na memória:

E também note que alguns caracteres como os parênteses são renderizados invertidos, isto porque eles tem um significado especial: Parênteses que abrem e fecham, e não parênteses esquerdos e direitos.

O funcionamento se baseia em caracteres fortes, fracos e neutros.

- Caracteres **fortes** tem direção definida (esquerda pra direita ou esquerda pra direita), geralmente são os caracteres restritos a um sistema de escrita, e são guardados na ordem lógica (da esquerda pra direita), mas renderizados de forma oposta.
- Caracteres **fracos** tem direção vaga, geralmente são os símbolos matemáticos e os números. Eles são guardados da esquerda para direita, e renderizados da esquerda para direita, porém apenas em blocos contíguos de caracteres fracos.
- Caracteres **neutros** tem direção indeterminada. Exemplos incluem separadores, espaços e sinais de pontuação comuns a vários sistemas de escrita. Visto que eles são separadores, quando estão nas bordas, não fazem parte do conjunto invertido para renderizar.

A divisão de anúncios da Elogog (empresa famosa da Nlogônia) está pensando em fazer um navegador de Internet e precisa da sua ajuda para certificar que os textos sejam renderizados corretamente!

Considere que caracteres **fortes** alteram a direção do texto da posição atual até o próximo caractere **forte** (ou o final do texto) e que caracteres **fracos** e **neutros** não alteram a direção do texto.

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 10^6$) que representa o tamanho da linha a ser convertida.

A segunda linha é de tamanho N com caracteres restritos a um subconjunto do conjunto ASCII. O alfabeto é composto por caracteres **neutros**: espaços simples, vírgulas, pontos, parênteses `()`, colchetes `[]`, chaves `{}` e parênteses angulares `<>`; caracteres **fracos**: números e operadores matemáticos `+`, `-`, `*`, `/`

e =; caracteres **fortes**: letras latinas minúsculas, que representam um texto em uma linguagem esquerda para direita, e letras latinas maiúsculas, que representam texto em uma linguagem direita para esquerda.

O primeiro caractere da linha é sempre um caractere **forte**.

Saída

Imprima uma linha de tamanho N na ordem de renderização.

Exemplos

entrada padrão
44 portugues 1 2 3 (UERBEH 1 234 (5 portugues
saída padrão
portugues 1 2 3 (5) 234 1 HEBREU portugues

entrada padrão
75 os SOD) SEBARA numeros SOCIBARA como 123), MAICINI da ->. LAUGI E 1234 SAM.
saída padrão
os ARABES (DOS numeros ARABICOS como 123), INICIAM da ->. MAS 1234 E IGUAL.

entrada padrão
79 equacoes [+54SETNEREFID OAS [100 = +12+34. mas ([ADNIA ([SALE, sao S}A)D]IL<AV.
saída padrão
equacoes [+54+12+34 = 100] SAO DIFERENTES. mas ([ELAS]) AINDA, sao VA>LI[D(A{S.

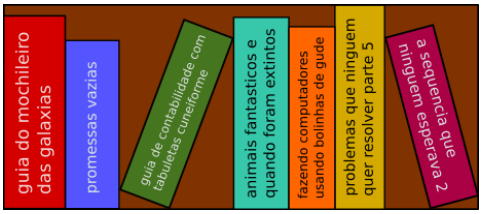
Problema B. Biblioteca Alfabetizada

Arquivo de entrada:	`entrada padrão`
Arquivo de saída:	`saída padrão`
Tempo limite:	1 segundo
Limite de memória:	256 megabytes

As bibliotecas públicas de Nlogônia estão um caos! Os livros estão todos fora do lugar nas prateleiras! Para nossa sorte, todos os sistemas são informatizados, então temos uma verdadeira bagunça organizada, que é terrível de se consultar manualmente, mas com o auxílio do sistema, é possível encontrar a localização de todos os livros.

O ministério dos livros não está muito feliz com essa situação, porque se o sistema parar de funcionar, ninguém sabe onde os livros estão. Pensando em resolver esse problema, o ministério então recrutou uma equipe de bibliotecários para resolver esse problema. O problema é que não se sabe quais lugares estão mais desorganizados que outros.

Para medir o trabalho que vai levar para organizar os livros de cada biblioteca, eles pensaram numa métrica que eles chamam de *grau de desorganização*. O grau de desorganização é dado pela quantidade de conjuntos de dois livros que estão na ordem errada.



O livro "problemas que ninguem quer resolver parte 5" por exemplo está fora de ordem apenas em relação aos livros "a sequencia que ninguem esperava 2" e "promessas vazias". Isto é contrastado por exemplo ao livro "a sequencia que ninguem esperava 2" que está fora de ordem em relação a todos os outros livros. Lembre-se que são conjuntos, então a relação {"a sequencia que ninguem esperava 2", "problemas que ninguem quer resolver parte 5"} é contada apenas uma vez. O *grau de desorganização* dessa estante é 15.

O ministério então precisa da sua ajuda para distribuir a equipe! Sabendo do *grau de desorganização* de cada biblioteca, distribua a equipe, dizendo quantos devem trabalhar em cada biblioteca. Cada biblioteca desorganizada inicialmente deve ter um bibliotecário, e em seguida, o restante deve ser distribuído conforme a proporção dada pelo grau de desorganização. Caso sobre bibliotecários, eles deverão ser distribuídos um a um em ordem para os lugares mais desorganizados, escolhendo no caso de empate os listados primeiro.

Entrada

A primeira linha é dada pelo inteiro N ($1 \leq N \leq 10^5$), a quantidade de bibliotecas e o inteiro K ($1 \leq K \leq 10^{10}$), o tamanho da equipe que deve ser dividida entre as bibliotecas. É garantido que K é maior ou igual ao número de bibliotecas desorganizadas.

Em seguida, cada biblioteca é descrita por uma linha com um inteiro L ($1 \leq L \leq 10^5$) seguida de L linhas, cada uma contendo a descrição de cada livro da biblioteca em sua ordem original na prateleira.

A descrição do livro é composta por um inteiro M ($1 \leq M \leq 60$) que determina o tamanho do nome do livro, um espaço e o nome do livro (que não possui espaços no começo ou no final). O nome dos livros possui apenas caracteres de espaço, números e o alfabeto latino minúsculo, que tem ordem lexicográfica definida nesta ordem.

É garantido que a soma de todos os M de todas as bibliotecas não excede 10^5.

Saída

Imprima uma linha para cada biblioteca na ordem listada na entrada, dizendo quantos bibliotecários

devem ser alocados para aquela biblioteca. A soma deve totalizar K ou 0 quando todas as bibliotecas estiverem organizadas.

Exemplos

entrada padrão
3 17 7 31 guia do mochileiro das galaxias 16 promessas vazias 46 guia de contabilidade com tabuletas cuneiforme 43 animais fantasticos e quando foram extintos 44 fazendo computadores usando bolinhas de gude 43 problemas que ninguem quer resolver parte 5 34 a sequencia que ninguem esperava 2 3 19 dicionario de s a z 19 dicionario de a a f 19 dicionario de g a l 4 31 computadores e outros problemas 33 discussoes sobre provas de np e p 24 prova que np e igual a p 28 prova que np nao e igual a p
saída padrão
15 2 0

entrada padrão

```
5 9
2
36 jeitos de criar habitos improdutivos
32 5 provas matematicas gigantescas
3
32 ate mais e obrigado pelos peixes
32 o restaurante no fim do universo
29 a vida o universo e tudo mais
3
11 the minds i
19 i am a strange loop
17 godel escher bach
4
60 problemas tipograficos causados pelo uso de titulos muito gr
60 a longa lista dos livros que tem titulos extremamente longos
58 problemas nao resolvidos da computacao que estao em aberto
60 lista de titulos tautologicos que tem titulos com tautologia
10
40 the art of computer programming volume 1
40 the art of computer programming volume 2
41 the art of computer programming volume 4d
40 the art of computer programming volume 3
41 the art of computer programming volume 4a
41 the art of computer programming volume 4b
40 the art of computer programming volume 5
41 the art of computer programming volume 4c
40 the art of computer programming volume 6
40 the art of computer programming volume 7
```

saída padrão

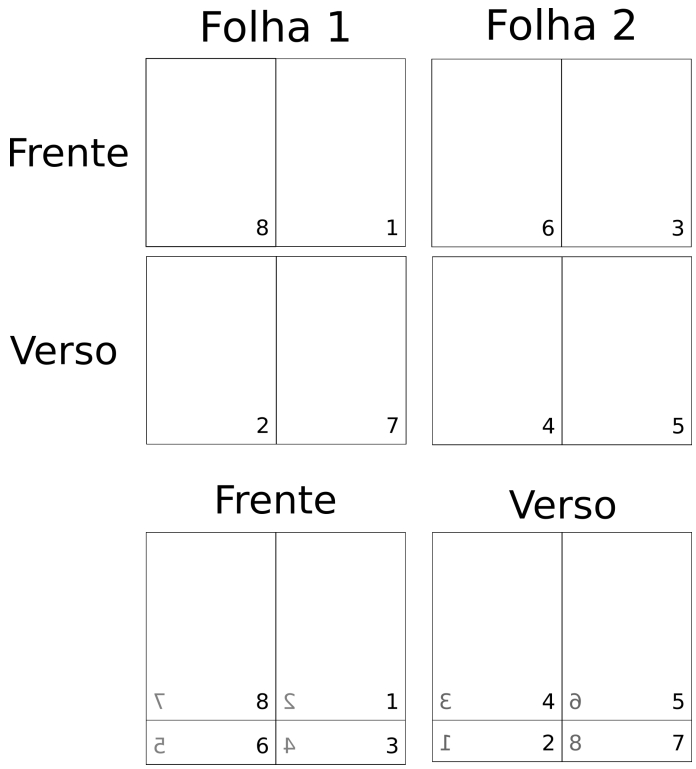
```
1
1
1
3
3
```

Problema C. Cordéis Incompletos

Arquivo de entrada:	`entrada padrão`
Arquivo de saída:	`saída padrão`
Tempo limite:	1 segundo
Limite de memória:	256 megabytes

A literatura de cordel é clássica na Nlogônia. Os textos são dos mais variados tópicos, sendo que as façanhas do cangaceiro LEDão são os textos que mais tiveram tiragem na história dos cordéis. Por ser um fenômeno de tamanha importância, ele hoje é considerado um patrimônio cultural imaterial da Nlogônia.

O cordel é publicado em um formato de bolso que tem tamanho de página entre 110mm × 150mm e 135mm × 180mm. Para chegar nessa formato de página, é pego uma folha de papel e realizado uma sequência de cortes e dobraduras. O formato que trabalharemos é o de 4 páginas de texto em cada folha, publicado em formato *folio*, com duas páginas em cada lado. Observe a figura:



Dentro da biblioteca pública do distrito de Loglogia, foram encontradas caixas com cordéis. Porém, ao fazer uma inspeção rápida, foi notado que alguns cordéis estavam com folhas faltando, e outros nem eram cordéis!

Sua missão é dado as dimensões dos papéis e dados os números de página encontrados em cada folha de papel, avaliar se os papéis dados correspondem a um cordel, e se sim, descrever quais folhas estão faltando.

Entrada

A primeira linha da entrada contém inteiros N e M ($1 \leq N, M \leq 10^4$), as dimensões em milímetros das folhas de papel encontradas na caixa.

Na próxima linha, segue um inteiro F ($1 \leq F \leq 10^5$), a quantidade de folhas de papel.

Em seguida, nas próximas F linhas, são descritos quatro números de página f_1, f_2, f_3, f_4 ($1 \leq f_i \leq 4 \cdot 10^5$) encontrados em cada folha de papel.

É garantido que todos os f_i são distintos e que eles correspondem a números de página em formato *folio*. É garantido também que a folha da capa sempre está presente.

Saída

Se o conteúdo da caixa não corresponder a um cordel, imprima `'NO'` em uma linha.

Se sim, imprima `'YES'` em uma linha, e na próxima linha, imprima o inteiro K, o número de folhas faltantes. Nas próximas K linhas, imprima em cada uma k_1, k_2, k_3, k_4 separados por espaço, os números de página referentes a cada papel faltante, em qualquer ordem.

Exemplos

entrada padrão	saída padrão
220 300 2 6 4 5 3 8 1 2 7	Y 0
300 220 1 1 8 7 2	YES 1 6 5 3 4
280 300 2 6 4 5 3 8 1 2 7	N

Problema D. Dígitos Verificadores

Arquivo de entrada:	`entrada padrão`
Arquivo de saída:	`saída padrão`
Tempo limite:	1 segundo
Limite de memória:	256 megabytes

O primeiro erro de matemática registrado na história da humanidade provavelmente foi cometido por um contador 5 milênios atrás cujo nome é Kushim, que é o ser humano mais antigo cujo nome se tem registro. Junto com seu supervisor, Nisa, Kushim cuidava das contas de um armazém, contando a quantidade de cevada armazenada, assim como o registro das ordens de cerveja e receitas com as razões de cevada necessárias para prepará-las. É possível, portanto, que o primeiro erro da matemática seja só resultado de tentar fazer matemática enquanto se está bêbado.

Alguns milênios depois, por volta do ano 950, um astrônomo indiano, Aryabhata II, descreveu o uso de um método para verificar somas, subtrações, multiplicações e divisões numéricas. Esse método é chamado de "prova dos noves". Ele consiste em pegar um número e tirar a sua soma digital (por exemplo, a soma digital de 2946 é $2 + 9 + 4 + 6 = 21$), e em seguida pegar esse número módulo 9 (o módulo de 21, por exemplo, é 3).

O método da "prova dos noves" é quase equivalente a chamada "raiz digital", que envolve pegar um número e repetir o processo de soma digital várias vezes (por exemplo, a soma digital de $21 = 2 + 1 = 3$). Elas só não são equivalentes em relação aos múltiplos de nove, pois a soma digital de um múltiplo de nove não-zero é 9, enquanto que o método da "prova dos noves" iria resultar em 0. De toda forma, ambas são equivalentes $\pmod 9$.

É possível provar que a "raiz digital" $\pmod 9$ e o método da "prova dos noves" $\pmod 9$ são na verdade equivalentes ao próprio número $\pmod 9$. Isso permite que as propriedades dos números modulares seja extrapolada para esses outros métodos, o que permite que somas, subtrações, multiplicações e divisões sejam verificadas:

$$dr(a_1 + a_2) = dr(dr(a) + dr(b))$$
$$dr(ab) = dr(dr(a)dr(b))$$

Isso significa que se fizermos uma operação de soma ou de multiplicação, podemos fazer a raiz digital $\pmod 9$ de forma separada nos operandos e na resposta, e o resultado tem que ser idêntico, temos então um dígito verificador.

Sabendo de tudo isso, vamos ajudar Nisa a achar os erros nas contas de Kushim usando o método dos "múltiplos de nove". Primeiro, transcrevemos nossas tabelas de multiplicação (assim como nos tablets de Kushim) e números iniciais numa fita contínua para facilitar o reuso dos números e vamos verificar se o resultado da nossa soma ou multiplicação resulta no mesmo dígito verificador que é esperado. Vamos então ir colocando nos tablets os resultados das somas, até chegarmos na resposta final!

Entrada

Na primeira linha, é dado o inteiro N ($1 \leq N \leq 10^5$), o tamanho da fita de números. A linha que vem a seguir contém a fita de números que é uma sequência de dígitos.

A próxima linha contém o inteiro Q ($1 \leq N \leq 10^5$), o número de operações que vamos realizar. Em seguida, seguem Q linhas, cada uma com uma operação. No começo de cada linha há um identificador de operação inteiro O:

- Para $O = 1$ ou $O = 2$, sendo que 1 representa soma e 2 representa multiplicação, temos dois inteiros A_1, B_1 ($1 \leq A_1 \leq B_1 \leq N$), dois inteiros A_2, B_2 ($1 \leq A_2 \leq B_2 \leq N$) e o dígito de verificação obtido D ($0 \leq D \leq 8$). Os inteiros A_1 e B_1 representam o início e o fim inclusivo do primeiro operando na fita, enquanto que A_2 e B_2 representam o início e o fim inclusivo do segundo operando na fita.

- Para $O = 3$, temos um inteiro P ($1 \leq P \leq N$) e um inteiro D ($0 \leq D \leq 9$) que representam uma posição na fita e um dígito que será colocado naquela posição. Qualquer dígito previamente naquela posição é apagado.

Saída

Para cada operação $O = 1$ ou $O = 2$, imprima `'YES'` se o dígito D informado bate com a raiz digital do resultado da operação $\pmod 9$, ou `'NO'` se não.

Exemplo

entrada padrão
48 819235303009000007061060012102700594252702953000 15 2 1 1 6 6 4 2 2 3 7 8 3 2 4 4 9 11 6 2 5 5 12 14 0 3 15 4 3 17 5 1 15 16 17 19 7 1 20 22 23 25 4 1 26 29 30 33 4 2 6 6 34 34 7 2 35 35 7 8 0 2 36 36 12 14 0 1 37 38 39 41 7 3 46 6 1 42 44 45 48 4

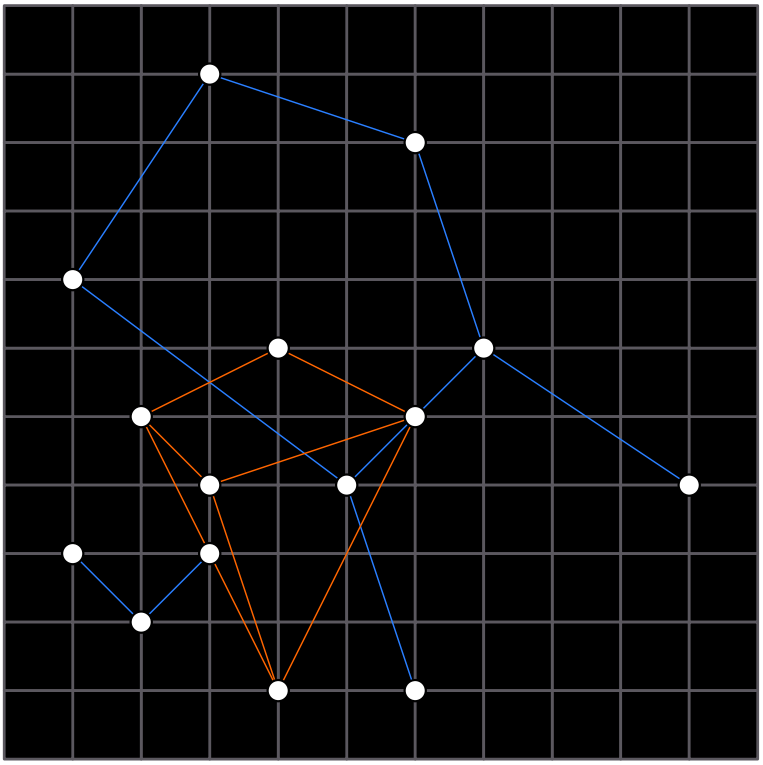
saída padrão
YES YES YES YES YES YES YES YES YES YES YES NO

Problema E. Estrelas Constelares

Arquivo de entrada:	`entrada padrão`
Arquivo de saída:	`saída padrão`
Tempo limite:	1 segundo
Limite de memória:	256 megabytes

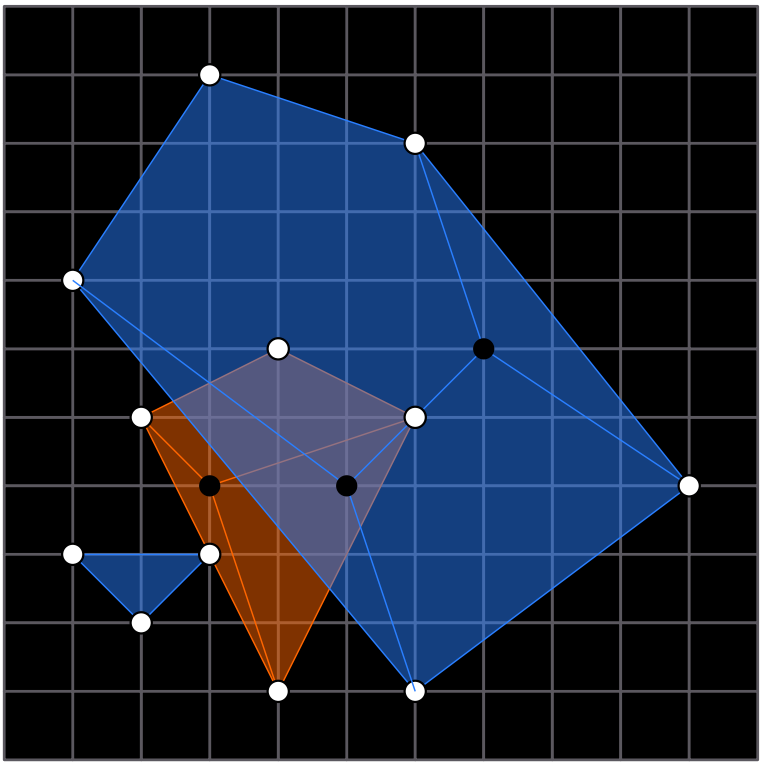
Desde o início da história da humanidade, as estrelas do céu noturno nos fascinam. Elas fazem parte das mitologias, dos nossos símbolos, e inspiraram o nosso fascínio em estudar o que existe além das estrelas. Hoje elas são usadas para navegação e para validar diversas teorias científicas sobre a natureza do universo.

Os nossos antepassados pré-históricos avistavam os pontos brilhantes no céu, e buscavam reconhecer padrões e formar figuras e desenhos ligando os pontos, figuras que hoje chamamos de constelações. Figuras de animais, criaturas mitológicas ou objetos inanimados são formadas ligando estrelas por segmentos imaginários que formam um desenho no céu.



Algumas estrelas, porém, contribuem mais com a forma geral da constelação do que outras. Chamamos essas estrelas de **estrelas constelantes**. Para descobrir exatamente quais contribuem mais para a forma geral da constelação, criamos um polígono que encobre toda a constelação e classificamos aquelas que estão nas bordas deste polígono como estrelas constelantes. Esse polígono não possui três pontos colineares.

Descubra então quais são as estrelas constelantes dadas as constelações, sendo elas aquelas que fazem parte da borda do polígono de pelo menos uma constelação.



Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 10^3$), a quantidade de constelações. Em seguida, seguem as descrições das constelações.

Cada constelação é definida por uma linha com um inteiro L ($1 \leq L \leq 500$) que diz quantos segmentos existem naquela constelação. Em seguida, L linhas seguem, cada uma descrevendo os segmentos da constelação, com inteiros A_x, A_y ($-10^5 \leq A_x, A_y \leq 10^5$), o ponto inicial do segmento e B_x, B_y ($-10^5 \leq B_x, B_y \leq 10^5$), o ponto final do segmento que é diferente do ponto inicial. Não há segmentos especificados mais de uma vez. É garantido que as constelações formam corpos conectados.

Saída

Imprima um inteiro C, o número de estrelas constelantes.

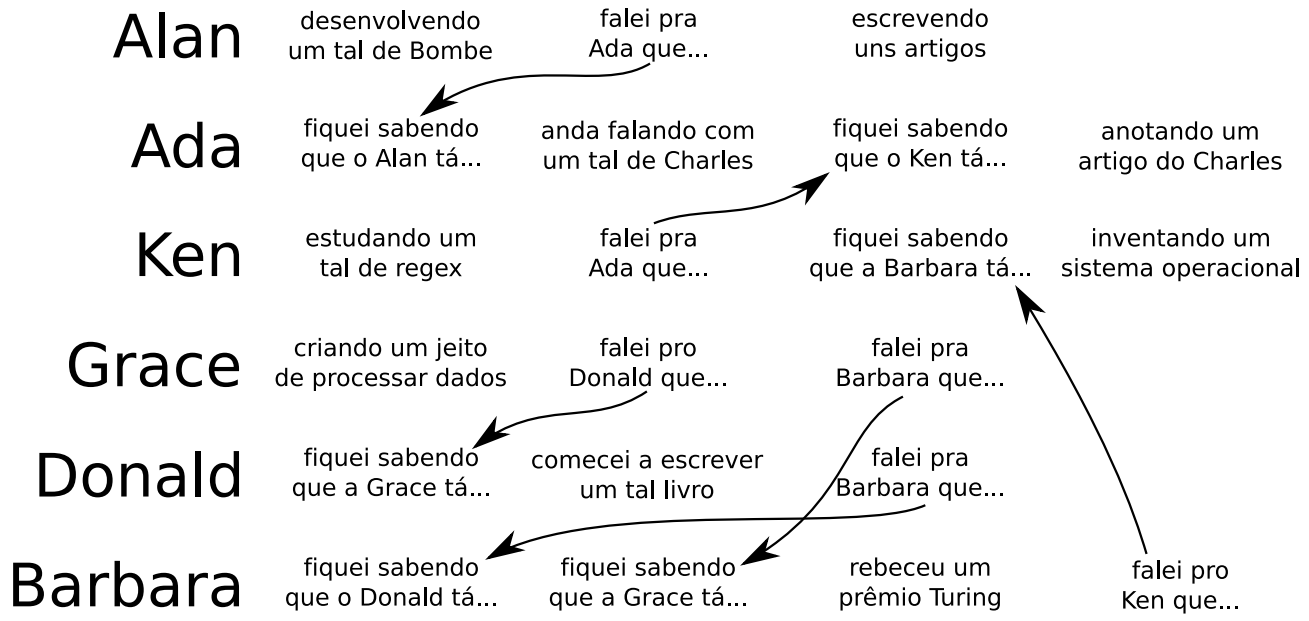
Exemplo

entrada padrão	saída padrão
3 8 1 3 2 6 2 6 5 7 7 10 5 7 5 7 6 6 7 5 6 6 4 1 7 5 4 1 1 3 7 5 10 6 8 7 3 10 4 6 2 5 4 6 6 10 4 8 3 6 2 7 3 6 6 5 4 6 6 6 2 7 3 8 3 10 4 2 9 2 8 3 9 2 8 1	12

Problema F. Fofoca Corre Solta

Arquivo de entrada:	`entrada padrão`
Arquivo de saída:	`saída padrão`
Tempo limite:	0.5 segundos
Limite de memória:	256 megabytes

Ariel adora uma fofoca. Dentro do seu grupo de amigos, Ariel sabe de absolutamente tudo que acontece. Todas as intrigas, e também quando alguém fofocou ou recebeu a fofoca de alguém. Porém, é muito complicado saber exatamente que ordem as coisas aconteceram. Ariel, sabendo de todas as fofocas, frequentemente se confunde em dizer que x aconteceu antes de y, quando na verdade era o contrário! Para resolver esse problema, Ariel então começou a registrar suas fofocas:



Assim, Ariel é capaz agora de saber exatamente o que aconteceu antes do que! Por exemplo, Ken ter ouvido a fofoca de Barbara que ela recebeu o prêmio Turing definitivamente aconteceu antes de ele contar que está inventando um sistema operacional. Em alguns casos, é impossível fazer qualquer correlação, como por exemplo Alan contar que está desenvolvendo um tal de Bombe não aconteceu nem antes nem depois de Grace contar que está criando um jeito de processar dados, mas não tem problema, conforme mais fofocas forem ocorrendo, essas incertezas sumirão.

Como Ariel está tendo que registrar uma quantidade absurda de fofocas, pediu sua ajuda para administrar essa lista. Pegue a lista de Ariel e responda suas perguntas quanto a quando a ordem de determinados eventos.

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 10^3$), a quantidade de amigos fofoqueiros de Ariel.

Seguem então N linhas, uma para cada amigo fofoqueiro i ($1 \leq i \leq N$). Nesta linha se encontra um inteiro M_i ($1 \leq M_i \leq 100$) e M_i inteiros separados por espaço m_j ($-N \leq m_j \leq N$), sendo que $m_j \neq i$ e $m_j \neq -i$. Cada inteiro m_j representa um tipo de fofoca. Fofocas de tipo 0 são novidades, enquanto que fofocas do tipo $m_j > 0$ revelam que o fofoqueiro i fofocou para m_j, enquanto que fofocas do tipo $m_j < 0$ revelam que o fofoqueiro i ouviu uma fofoca de $-m_j$.

É garantido que as fofocas são consistentes, isto é, cada par de fofocas m_j e $-m_j$ é inserido de forma conjunta no final de ambas as listas dos participantes da fofoca.

Em seguida, há uma linha com um inteiro Q ($1 \leq Q \leq 100$) que indica a quantidade de perguntas que Ariel quer fazer. As próximas Q linhas são compostas dos inteiros i ($1 \leq i \leq N$) e f_i ($1 \leq f_i \leq M_i$), j ($1 \leq j \leq N$) e f_j ($1 \leq f_j \leq M_j$) com $(i, f_i) \neq (j, f_j)$.

Saída

Para cada pergunta de Ariel,

- Imprima `<` em uma linha caso a fofoca de índice f_i do fofoqueiro i aconteceu antes da fofoca de índice f_j do fofoqueiro j
- Imprima `>` em uma linha caso a fofoca de índice f_j do fofoqueiro j aconteceu antes da fofoca de índice f_i do fofoqueiro i
- Imprima `?` em uma linha se não for possível dizer se uma fofoca aconteceu antes da outra.

Exemplo

entrada padrão	saída padrão
6 3 0 2 0 4 -1 0 -3 0 4 0 2 -6 0 3 0 5 6 3 -4 0 6 4 -5 -4 0 3 6 3 3 3 4 3 4 3 3 1 1 4 1 3 3 4 1 6 4 3 4 3 4 2 4	< > ? > < ?

Problema G. Glob Só Que Louco

Arquivo de entrada:	`entrada padrão`
Arquivo de saída:	`saída padrão`
Tempo limite:	0.25 segundos
Limite de memória:	256 megabytes

O comando `glob`, nome curto para `global`, era um programa presente nas primeiras versões do Unix. Sua função era fazer expansão de uma expressão e passar para outro programa todos os arquivos que casavam com aquele expressão. Como exemplo, a expressão `‘*’` casa com o nome de todos os arquivos, então pode literalmente significar "todos os arquivos". Nos dias atuais, o `glob` foi englobado por funções de biblioteca, e faz parte da habilidade padrão dos *shells*.

Porém, Zaphod não trabalha com arquivos no seu dia a dia. Seu trabalho está ligado a criação de um Sistema Gerenciamento de Banco de Dados (SGBD) chamado MyPostcle. Nesse banco de dados, é usado uma linguagem de consulta Só Que Louco (SQL), que permite consultas nesse estilo:

```
SELECIONE nome, salario DE funcionario ONDE nome EH TIPO ‘Zaphod B%’ E salario EH TIPO
‘[1-9]__[34][^9][^8]’
```

Agora, está na fase do projeto de implementar a funcionalidade do `TIPO` que é extremamente parecida com os globs do Unix. Ao invés de usar `‘*’` e `‘?’`, a sintaxe é um pouco diferente:

- `‘%’`: Casa com uma sequência de 0 ou mais caracteres
- `‘_’`: Casa apenas com um caractere qualquer
- `‘[A-Zd-w1]’`: Casa com todas as letras do intervalo `‘A’` a `‘Z’`, `‘d’` a `‘w’` ou o caractere `‘1’`.
- `‘[^AW]’`: Casa com todos os caracteres exceto `‘A’` e `‘W’`.

O Zadoph fez um algoritmo, mas ele não passou em todos os testes do SGBD. Desesperado, ele agora está recorrendo a você para ajudá-lo, você consegue implementar a funcionalidade de `TIPO` da linguagem Só Que Louco?

Entrada

Na primeira linha se encontra um inteiro N ($1 \leq N \leq 100$), o tamanho do padrão `TIPO`. Na próxima linha, se encontra o padrão de tamanho N que é composto de caracteres ASCII de espaço, alfabeto latino maiúsculo e minúsculo, números, `‘[’`, `‘]’`, `‘^’`, `‘-’`, `‘%’` e `‘_’`.

É garantido que depois de `‘[’`, opcionalmente existe um `‘^’`, e sempre existe um conjunto de caracteres não especiais que podem ser caracteres isolados ou intervalos `‘A-Z’` que tem extremidades pertencentes ao mesmo grupo (maiúsculas com maiúsculas, minúsculas com minúsculas, números com números, espaço com espaço), e finalmente é sempre finalizado por `‘]’`. Os caracteres `‘[’`, `‘]’`, `‘-’` e `‘^’` não são usados fora deste contexto.

Na linha seguinte, se encontra um inteiro Q ($1 \leq Q \leq 10^3$), a quantidade de linhas que serão testadas. Nas Q linhas seguintes, se encontra um inteiro M ($1 \leq M \leq 100$) que determina o tamanho do texto a ser testado, um espaço, e o texto a ser testado (que não possui espaços no começo ou no final). O texto possui apenas caracteres de espaço, do alfabeto latino maiúsculo e minúsculos e números.

Saída

Imprima todos as linhas que casaram com o padrão na ordem que foram dadas na entrada.

Exemplos

entrada padrão	saída padrão
10 Z%aph_d B% 11 17 Zaphod Beeblebrox 20 Ziayphad Bittersweet 21 Ziyaphad Betelgeusian 21 Tricia Marie McMillan 20 Zaphod Beeblebrox II 12 Ford Prefect 23 Zetaphod Beeblebrox III 21 Z123aphad Bittersweet 18 Arthur Philip Dent 20 Zetaphad Bittersweet 21 Z123phad Betelgeusian	Zaphod Beeblebrox Ziyaphad Betelgeusian Zaphod Beeblebrox II Zetaphod Beeblebrox III Z123aphad Bittersweet Zetaphad Bittersweet
19 [1-9]__[34][^9][^8] 8 6 100387 6 100487 6 100388 6 900387 7 9003387 6 a03387 6 103487 6 103587	100387 100487 900387 103487

Problema H. Horas Nlogônicas

Arquivo de entrada:	`entrada padrão`
Arquivo de saída:	`saída padrão`
Tempo limite:	1 segundo
Limite de memória:	256 megabytes

Nlogônia acabou de passar por uma revolução. A Tomada da Pastilha, que ocorreu exatamente às 09:45:45, deu início a um movimento que quer retirar os sistemas de medição imperiais. Então agora os revolucionários estão decidindo um novo sistema de medição de tempo, e visto que nos relógios analógico-digitais, a Tomada ocorreu em um momento de "ponteiros iguais", foi estabelecido que um critério importante para a escolha de um novo sistema de medição de tempo seria comparar os horários em que existem "ponteiros iguais" no relógio.

Vale lembrar que os relógios analógicos-digitais são um tipo de relógio analógico popular na Nlogônia, onde os ponteiros movem de maneira quantizada comandada por um sistema digital. Isso significa que por exemplo, o ponteiro de uma hora só se move 12 vezes, 30 graus ao final de cada hora, sem pontos intermediários.

Um sistema famoso proposto é o sistema de tempo decimal. Nele, um dia é composto de 10 horas, 100 minutos por hora e 100 segundos por minuto. Nele, os "ponteiros iguais" ocorrem dez vezes, uma em 00:00:00, outra em 01:10:10, e assim por diante. Porém, esse sistema é considerado ambíguo, pois 100 é usado para minutos e segundos, mas ele ainda pode ser considerado caso esses conjuntos de "ponteiros iguais" sejam mais elegantes que os outros.

Os revolucionários então, querem saber exatamente quando que ocorrem "ponteiros iguais" em relógios analógicos-digitais para cada sistema de medição de tempo dado.

Entrada

São dados três números inteiros em uma linha: H ($1 \leq H \leq 10^6$), a quantidade de horas, M ($1 \leq M \leq 10^6$), a quantidade de minutos em um hora e S ($1 \leq S \leq 10^6$), a quantidade de segundos em um minuto.

Saída

Imprima uma linha com inteiro K, a quantidade de "ponteiros iguais" existentes. Em seguida, imprima K linhas, cada uma com os inteiros H, M e S separados por espaço correspondendo aos horários em que ocorrem cada "ponteiros iguais".

Exemplos

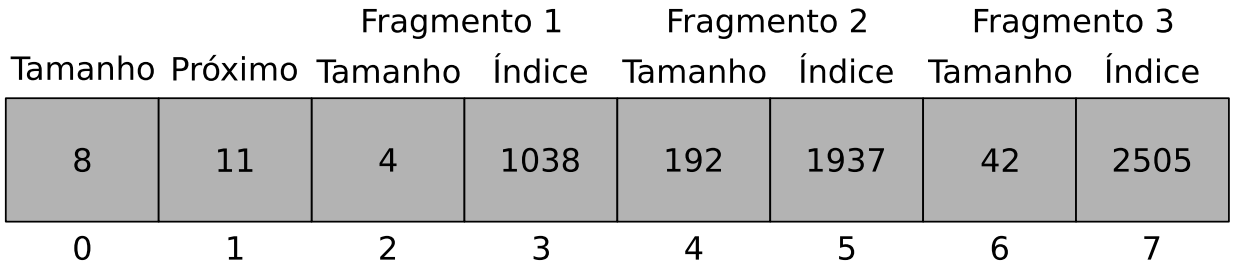
entrada padrão	saída padrão
10 100 100	10 0 0 0 1 10 10 2 20 20 3 30 30 4 40 40 5 50 50 6 60 60 7 70 70 8 80 80 9 90 90
12 60 60	12 0 0 0 1 5 5 2 10 10 3 15 15 4 20 20 5 25 25 6 30 30 7 35 35 8 40 40 9 45 45 10 50 50 11 55 55

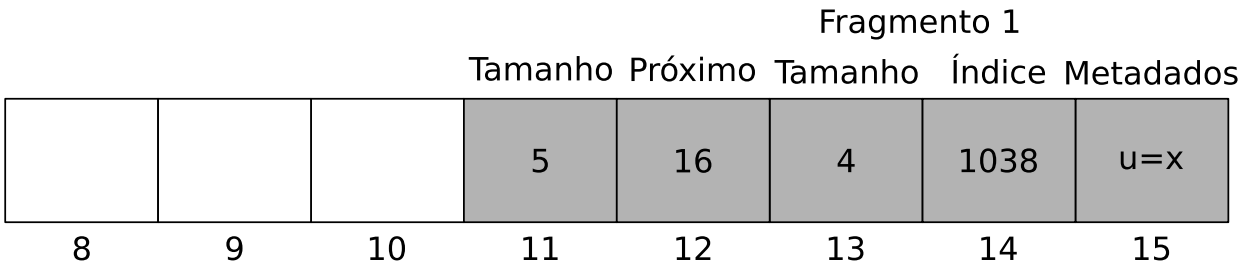
Problema I. Índice de Arquivos

Arquivo de entrada:	`entrada padrão`
Arquivo de saída:	`saída padrão`
Tempo limite:	1 segundo
Limite de memória:	256 megabytes

O sistema de arquivos Fila de Arquivos no Topo (FAT) possui uma lista ligada de arquivos no topo. Arquivos e as entradas de pasta são guardados em alguma localização do disco e para controle do sistema de arquivos, é criado um nó ao final da lista ligada no topo que representa aquele arquivo ou pasta.

Em cada nó dessa lista, é encontrado os índices do começo dos fragmentos do arquivo no disco, o tamanho desses fragmentos, opcionalmente alguns metadados, e o índice do próximo nó dessa lista ligada.





Este sistema de arquivos tem esse nome porque permite que possíveis becapes do sistema sejam facilmente criados. Qualquer modificação no conteúdo dos arquivos ou nos seus metadados implica a deleção daquela entrada, e ela é inserida novamente no fim da fila. Os criadores do sistema de arquivos sabiam que os espaços deixados iam ser problemáticos, e implementaram um algoritmo simples que detectava se tinha espaços no arredor do final da fila e moviam algumas entradas pra trás.

Esse algoritmo funciona bem para o usuário comum que não cria e deleta uma quantidade excessivas de arquivos. Mas os usuários ferrenhos de computadores que se vangloriam com seus *uptimes* gloriosos, começaram a ver que isso é um problema sério, pois o espaço para guardar arquivos no disco ficava cada vez menor, afinal não se pode guardar arquivos no meio dessa fila de arquivos no topo.

No mercado, já existia uma ferramenta similar, o *Desfragmentaitor 3000*, que organiza a posição dos arquivos de forma a melhorar o desempenho da máquina, isto é, diminuir os espaços que estão entre os arquivos apontados pela fila de arquivos no topo e unificar entradas. Porém, isso não resolve o nosso problema, pois queremos otimizar os espaços encontrados nessa fila, ao invés dos arquivos apontados pela fila.

Então, como usuário ferrenho, você decidiu criar uma solução para desfragmentar a fila de arquivos no topo. O seu programa funcionou muito bem, mas ele é excessivamente lento, porque ele move todos as entradas da fila umas próximas das outras, e no seu sistema de arquivos com mais de 120 000 arquivos, isso demora um tempo considerável.

Então, você teve uma ideia: Limitar o programa para uma quantidade de tempo. O tempo para mover uma entrada da fila de arquivos é proporcional ao tamanho das entradas a serem movidas. Então, bole um algoritmo que libere a maior quantidade de espaço no disco (lembrando que o espaço no disco começa

no final da fila) dado um tempo para sua execução. Lembre-se, porém, de manter a propriedade temporal do sistema de arquivos.

Entrada

Na primeira linha, se encontra um inteiro N ($1 \leq N \leq 10^6$), a quantidade de arquivos ou pastas do sistema.

Em seguida, seguem N linhas, cada uma descrevendo uma entrada da lista ligada. As linhas contém M ($4 \leq M \leq 10^8$), o tamanho da entrada e P ($0 \leq P \leq 10^{10}$), a posição do próximo item da lista no disco, sendo que o último aponta para o primeiro.

Os itens são dados em ordem de posição no disco que começa na posição 0, onde P é estritamente crescente e sempre aponta para a entrada na próxima linha (exceto pelo último). As inconsistências na lista já foram detectadas e arrumadas nesta etapa do processo, então é garantido que o sistema de arquivos é consistente.

Por fim, é dado em uma linha o tempo T ($0 \leq T \leq 10^8$), que é o tempo disponível para mover os arquivos, medido em quantidade de posições de memória que podem ser movidas.

Saída

Imprima um inteiro K, o máximo de espaço que é possível liberar em tempo T.

Exemplos

entrada padrão	saída padrão
3 8 11 5 16 6 0 11	3
3 5 6 4 11 6 0 9	1