Universidade Federal de Goiás – UFG
Instituto de Informática – INF
Bacharelado em Sistemas de Informação

Algoritmos e Estruturas de Dados 1 – 2018/2

Lista de Exercícios nº 08 – Listas Lineares, Filas e Pilhas
(Turmas: INF0286, INF0061 – Prof. Wanderley de Souza Alencar)

Sumário

Observação: Todos os exercícios desta lista devem ser resolvidos utilizando listas lineares encadeadas ou suas variantes: *pilhas* e *filas*.

1 Fila do SUS

 (++)

Os pacientes que chegam na fila do *Sistema Único de Saúde* (SUS) passam por uma "triagem" e, imediatamente, vão para a fila de atendimento [1]. Na triagem a enfermeira anota o horário de entrada do paciente e quantos minutos ele tem até que sua condição de saúde se torne crítica[2].
Sabe-se que os pacientes são atendidos de 30 em 30 minutos, ou seja, sempre nas horas "cheias" ou em "meias horas", quando na fila de atendimento. O início da triagem e do atendimento se dá, pontualmente, às 7h da manhã. Se não há nenhum paciente sendo atendido e a fila está vazia[3], o primeiro paciente é atendido no instante que chega na triagem. O médico atende até o último paciente na fila[4]. A preocupação é se algum paciente atingiu uma condição crítica enquanto não tenha sido atendido[5]. Para tanto você foi convidado para verificar na fila quantos pacientes atingem a condição crítica.

Entrada

A entrada começa com uma linha com o número inteiro $n \in \mathbb{N}^*$, $0 < n < 25$, o número de pacientes que chegam à triagem. A seguir são n linhas com os valores inteiros $h, m, c \in \mathbb{N}^*$ e $7 < h < 19$, $0 \leq m < 60$ e $0 \leq c \leq 720$. Os valores de h e m correspondem, respectivamente, à hora e ao minuto em que o paciente chega à triagem. O paciente da linha i sempre chega antes que, e no máximo junto com, o paciente da linha $(i+1)$. O valor c é tempo, em minutos, antes do paciente atingir a condição crítica de saúde.

Saída

Imprima o número de pacientes que atingiram a condição crítica ainda na fila de atendimento.

Exemplos

[1] Lembre-se: Isto é apenas um exercício de programação e, portanto, não há nenhuma fidedignidade com os acontecimentos reais.
[2] Sim, ela é uma enfermeira que, nas horas vagas, opera uma "*bola de cristal*" dos contos infantis.
[3] Isto existe?
[4] Tá de brincadeira!
[5] Tá de brincadeira, de novo!

Entrada	Saída
4 7 0 20 7 0 30 7 30 20 8 15 30	1

Entrada	Saída
5 10 20 50 10 30 30 11 10 20 12 0 0 12 10 30	0

Entrada	Saída
24 7 0 0 7 0 30 7 0 60 7 0 90 7 0 120 7 0 150 7 0 180 7 0 210 7 0 240 7 0 270 7 0 300 7 0 330 7 0 360 7 0 390 7 0 420 7 0 450 7 0 480 7 0 510 7 0 540 7 0 570 7 0 600 7 0 630 7 0 660 7 0 690	0

Entrada	Saída
15 7 0 0 7 0 20 7 30 50 7 50 30 8 0 120 8 1 75 8 30 90 9 59 30 11 0 240 11 15 0 11 30 300 14 0 0 14 40 5 15 15 5 17 50 5	8

2 Ciranda, cirandinha, vamos todos cirandar!

 (++++)

No primeiro dia de um acampamento, devido à forte chuva, as atividades recreativas ficaram limitadas e as crianças foram levadas para o ginásio de esportes. Foi realizada uma gincana e uma das atividades da mesma consistiu em agrupar as crianças em um círculo (organizado no sentido anti-horário) do qual seriam retiradas, uma a uma, até que sobrasse apenas uma criança, que seria a vencedora.

No momento em que entra no círculo, cada criança recebe uma pequena ficha que contém um valor de 1 a 500. Depois que o círculo é formado, conta-se, iniciando na criança que está ao lado da primeira que entrou no círculo, o número correspondente à ficha que a primeira detém. A criança onde o número contado cair deve ser retirada do grupo, e a contagem inicia novamente da primeira criança que entrou, segundo o número da ficha da criança que acabou de ser eliminada.

Para ficar mais interessante, quando o valor que consta na ficha é *par*, a contagem é feita no sentido horário e quando o valor que consta na ficha é *ímpar*, a contagem é feita no sentido anti-horário[6].

Se a criança que entrou primeiro for retirada, assume seu posto a criança que entrou imediatamente depois (ou seja, a segunda criança que entrou), se esta também for retirada, assume aquela que entrou imediatamente depois dela (ou seja, a terceira criança que entrou), e assim por diante.

A brincadeira fez muito sucesso e o administrador do acampamento pediu para que sua equipe desenvolva um programa para que no próximo evento ele saiba, previamente, qual criança será vencedora de cada grupo, com base nas informações fornecidas.

Entrada

A primeira linha de cada caso de teste contém um inteiro n, com $(1 \leq n \leq 100)$, indicando a quantidade de crianças que farão parte de cada círculo e participarão da brincadeira. Em seguida, as n linhas seguintes, de cada caso de teste, conterão duas informações: o `nome` e o `valor` $(1 \leq valor \leq 500)$ que consta na ficha de cada criança, separados por um espaço, na ordem de entrada na formação do círculo.

Observação: O `nome` de cada criança não deverá ultrapassar 30 caracteres, sendo utilizado o caractere *sublinha* ("_") para nomes que contenham espaço em sua formação.

[6]Note que como a contagem começa do que está ao lado, isso muda se for *par* ou *ímpar*. Para *par*, a imediatamente ao lado no sentido horário, e para *ímpar*, a imediatamente ao lado no sentido anti-horário.

Saída

O programa deve apresentar, numa única linha, o nome da criança do grupo que venceu aquela brincadeira.

Exemplos

Entrada	Saída
3 Fernanda_Lima 7 Fernando_Lima 9 Gustavo_Lima 11	Fernanda_Lima

Entrada	Saída
5 Maria_Braga 7 Pedro_Machado 9 Joao_Newmann 5 Isabel_Cruz 12 Laura_Bras 8	Isabel_Cruz

Entrada	Saída
3 Maria_Braga 4 Pedro_Machado 3 Joao_Newmann 2	Pedro_Machado

Entrada	Saída
5 Ada_Lovelace 7 Alan_Turing 9 John_Newmann 5 Carow_Shaw 12 Frances_Allen 8	Carow_Shaw

3 A COPA do Mundo é Nossa...

🤓 (+)

Sempre que se aproxima a realização de uma *Copa do Mundo de Futebol*, o fluxo de pessoas nas filas para compra de ingressos aumenta consideravelmente. Quando as filas vão se tornando cada vez maiores, as pessoas menos pacientes tendem a desistir da compra de ingressos e acabam deixando as filas, liberando assim sua vaga para outras pessoas, o que é muito bom para os mais pacientes.

Quando uma pessoa deixa a fila, todas as pessoas que estavam atrás dela dão um passo à frente, sendo assim nunca existe um espaço vago entre duas pessoas consecutivas na fila. A fila inicialmente contém n pessoas $(1 \leq n \leq 60000)$, cada uma com um identificador diferente: m_i, $m_i \in \mathbb{N}^*$ e $1 \leq i \leq 60000$. Estes identificadores são, vulgarmente, chamados de "*senhas*".

Joãozinho, um menino que sagrou-se campeão da etapa regional goiana da OBI (Olimpíada Brasileira de Informática) coordenada pelo Prof. Wellington Martins do INF/UFG, sabe o *estado inicial* dela e os identificadores em ordem das pessoas que deixaram a fila. Ele também sabe que após o *estado inicial* nenhuma pessoa entrou mais na fila e deseja saber qual será o *estado final* desta fila.

Joãozinho não conhece "Estruturas de Dados" e, sabendo que você está cursando esta disciplina no INF/UFG, pediu sua ajuda para elaborar um programa de computador que resolva o problema.

Entrada

A primeira linha contém um inteiro n representando a quantidade de pessoas inicialmente na fila.

A segunda linha contém n inteiros estritamente positivos representando os identificadores m_i das pessoas na fila, com $1 \leq i \leq n$. O primeiro identificador corresponde ao identificador da primeira pessoa na fila, o segundo identificador da segunda pessoa na fila e assim sucessivamente. É garantido que duas pessoas diferentes não possuem o mesmo identificador.

A terceira linha contém um inteiro s, $(1 \leq s \leq n)$ representando a quantidade de pessoas que deixaram a fila.

A quarta linha contém s inteiros – s_i – estritamente positivos, $1 \leq s_i \leq 10^5$, representando os identificadores das pessoas que deixaram a fila, na ordem em que elas sairam. Da mesma maneira, é garantido que um mesmo identificador não aparece duas vezes nessa lista de pessoas que abandonaram a fila.

Saída

Seu programa deve imprimir uma linha contendo $(n-m)$ inteiros, correspondendo aos identificadores das pessoas que *permaneceram* na fila, em ordem de chegada a ela.

Exemplo

Entrada	Saída
8 5 100 9 81 70 33 2 1000 3 9 33 5	100 81 70 2 1000

Entrada	Saída
4 10 9 6 3 1 3	10 9 6

Entrada	Saída
8 10 2 3 4 66 45 32 77 1 3 10 2 3	4 66 45 32 77

Entrada	Saída
8 10 2 3 4 66 45 32 77 1 5 10 3 45 32 77	2 4 66

4 Branca de Neve e os n anões

 (++)

Branca de Neve e os n anões $\left(3 \leq n \leq 3 \times 10^5\right)$ vivem na floresta. Todas as manhãs, os anões formam uma longa fila indiana e vão assobiando para a mina. Branca de Neve corre ao redor deles e faz um monte de fotografias digitais – ela é uma menina tecnológica, que acessa e atualiza diariamente as suas redes sociais – para fazer o *upload* para a sua rede social favorita.

Quando os anões entram na mina, Branca de Neve volta para casa e passa a selecionar as fotos mais bonitas. Cada anão tem uma touca colorida, e há c cores diferentes disponíveis pela eles $(1 \leq c \leq 10000)$. Ela tem um gosto pitoresco: para ela, uma "foto bonita" é aquela em que mais da metade das toucas dos anões são da mesma cor[7]. Os anões não sabem disso e, por isso, não são capazes de escolher as suas toucas com a intenção de, deliberadamente, agradar Branca de Neve. Cada um faz sua escolha de maneira totalmente pessoal e desconhecida dos demais.

Você, um discípulo de Malba Tahan[8], quer verificar, para um conjunto de m fotos $(1 \leq m \leq 10000)$, quantas são "fotos bonitas" e qual a cor predominante.

Como você não quer fazer isto manualmente, resolveu escrever um programa de computador em $\mathbb{C}$ para cumprir a tarefa.

Entrada

A primeira linha da entrada conterá um inteiro t, o número de casos de teste, que virão na sequência, sabendo que $1 \leq t \leq 1000$ e que cada caso de teste possui $(3+m)$ linhas.

A primeira linha de cada caso de teste conterá os dois inteiros n e c, conforme anteriormente especificado.

A linha seguinte conterá n inteiros: $c_1, c_2, c_3, \ldots, c_n$, cada um correspondendo à cor da touca do respectivo anão, ordenados segundo a fila que eles formaram naquela manhã. É claro que $1 \leq c_i \leq c \leq 10000$.

A terceira linha de cada caso de teste conterá o valor de m – número de fotos registradas naquela manhã.

As m linhas seguintes conterão, cada uma, dois inteiros a e b, de tal maneira que $(1 \leq a \leq b \leq n)$. Cada uma destas linhas descreve uma foto, a qual contém todos os anões a partir do a-ésimo até b-ésimo, ou seja, não é obrigatório que todos os anões apareçam em todas as fotos registradas pela manhã.

[7] Explica-se: se houver k anões numa foto, ela é uma "foto bonita" se estritamente mais que $k/2$ anões usam toucas de mesma cor.

[8] Veja em `https://pt.wikipedia.org/wiki/Malba_Tahan`.

Saída

A saída deverá conter, para cada um dos t casos de teste, m mensagens (uma por foto registrada para aquele caso de teste).
Se Branca de Neve a considerou a foto uma "foto bonita", a mensagem deve ser "Sim X", onde X é a cor predominante nas toucas dos anões naquela foto[9]. Do contrário, a mensagem deve ser "Nao" (sem acento mesmo.)

[9]Lembre-se: em nosso caso a cor é representada por um número inteiro estritamente positivo.

Exemplos

Entrada	Saída
1 10 3 1 2 1 2 1 2 3 2 3 3 8 1 2 1 3 1 4 1 5 2 5 2 6 6 9 7 10	Nao Sim 1 Nao Sim 1 Nao Sim 2 Nao Sim 3

Entrada	Saída
2 10 3 1 2 1 2 1 2 3 2 3 3 8 1 2 1 3 1 4 1 5 2 5 2 6 6 9 7 10 12 5 1 5 2 3 2 4 2 5 3 4 2 4 6 1 5 2 7 9 12 3 12 4 9 1 12	Nao Sim 1 Nao Sim 1 Nao Sim 2 Nao Sim 3 Nao Nao Nao Nao Nao Nao

5 *Programming Strategy*

 (++)

The friends John, Jack and Joseph like to participate of programming competitions. They have differents strategy we would like to know which strategy is best:

John – simply solves the problems in the order in which he receives them from the contest organizers;

Jack – first reads all problems and solves them in increasing order of difficulty;

Joseph – also he reads all problems first, but he is very ambitious and, therefore, solves the problems in decreasing order of difficulty.

The *difficulty* of a problem is measured in "how many minutes" the boys take to solve the problem
We gathered statistics and consulted the "Thierson Couto Oracle" – an oracle as famous in INF/UFG as that of *Oracle of Delphi*, in Ancient Greece. So we know, for all types of problems, how much time the boys will need. We also found that for each problem, the three guys always need the same time, which depends on the difficulty of the problem. Thus, they differ just for their particular strategies.
For various competitions, we would like you to tell us the "winner", the number of problems solved and what your "score".
The "score" for a single problem is the time, in minutes, from the start of the concur until its resolution. The *overall scoreboard* is the sum of the scores of the problems solved.
The boys never make mistakes, so you do not have to deal with penalties. The "winner" is that that solve more problems and, in case of a tie, the one with the lowest score. If there is still tie, the three boys agree that José is the winner because he always brings delicious apple pies to everyone during workouts.

Input

The first line of the entry will contain an integer t, the number of test cases, where $1 \leq t \leq 1000$.
Each test case describes a contest and its first line tells how long the competition lasts d, in minutes, where $30 \leq d \leq 1440$, and number of problems p, from 3 to 24. In a second line you will have the "difficulties of problems", such as explained above, they say how many minutes (from 1 to 600) the guys need to solve the problems.

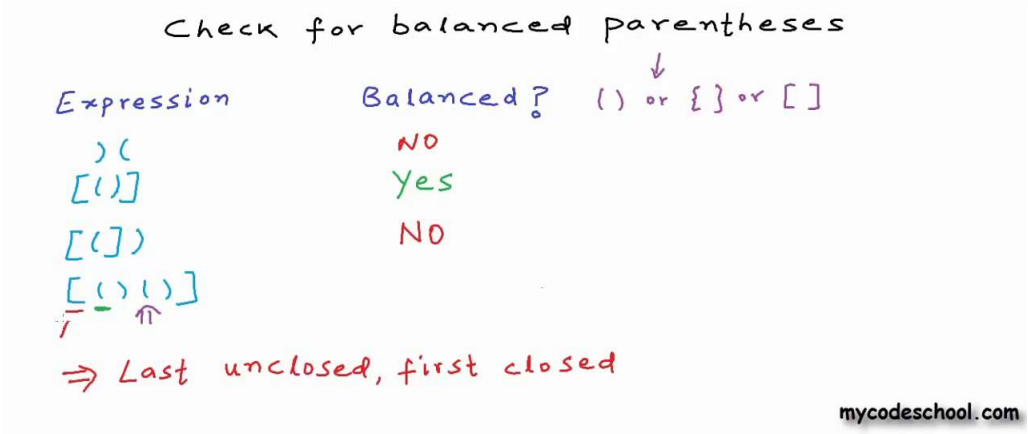
Output

The output must contain, for each test case, a line containing "Scenario i", where *i* is the number of the test case. Then a single line should be printed stating who is the "winner", the number of problems that he solves and, finally, his score.
Use the exact format as shown below in the example, even if the winner only solves 0 (zero) or 1 (one) problem. Complete the output of each test case with a blank line.

Example

Entrada	Saída
2 180 6 23 42 170 33 7 19 60 2 43 17	Scenario 1 Jack wins: 5 problems, 288 points. Scenario 2 Jack wins: 2 problems, 77 points.

Observation: Remember of the "commas" and "final dot" in the output (a single line each scenario).



6 Balanceamento de Parênteses

 (++)

Considere que seja dada uma expressão aritmética e qualquer, onde nela pode haver a presença de parênteses – (ou).
Por exemplo:

$$a*b-(2+c)$$

é uma expressão correta, pois há um "abre" parênteses e um "fecha" parênteses. O mesmo ocorre com:

$$(a+b*(2-c)-2+a)*2$$

Por outro lado, a expressão:

$$(a*b-(2+c)$$

está incorreta. Como também estão:

$$2*(3-a))$$

e

$$)3+b*(2-c)($$

Sintetizando: todo parênteses que "fecha" deve ter um outro parênteses que "abre" correspondente, bem como não pode haver parênteses que "fecha" sem um prévio parênteses que "abre". A quantidade total de parênteses que "abre" e "fecha" deve ser igual.
Você deve elaborar um programa de computador que seja capaz de verificar se uma dada expressão e, fornecida como entrada, está correta.
Observação: É obrigatório que seu programa utilize uma estrutura de dados "dinâmica" para implementar a solução.

Entrada

Uma única expressão e, com no mínimo 1 (um) caractere e no máximo 1000 (mil) caracteres, na linha de entrada.

Saída

Seu programa deve imprimir, numa única linha, a mensagem "Correta" ou "Incorreta". A análise deve ser realizada sem considerar o restante da expressão, apenas a relação existente entre os parênteses.

Exemplos

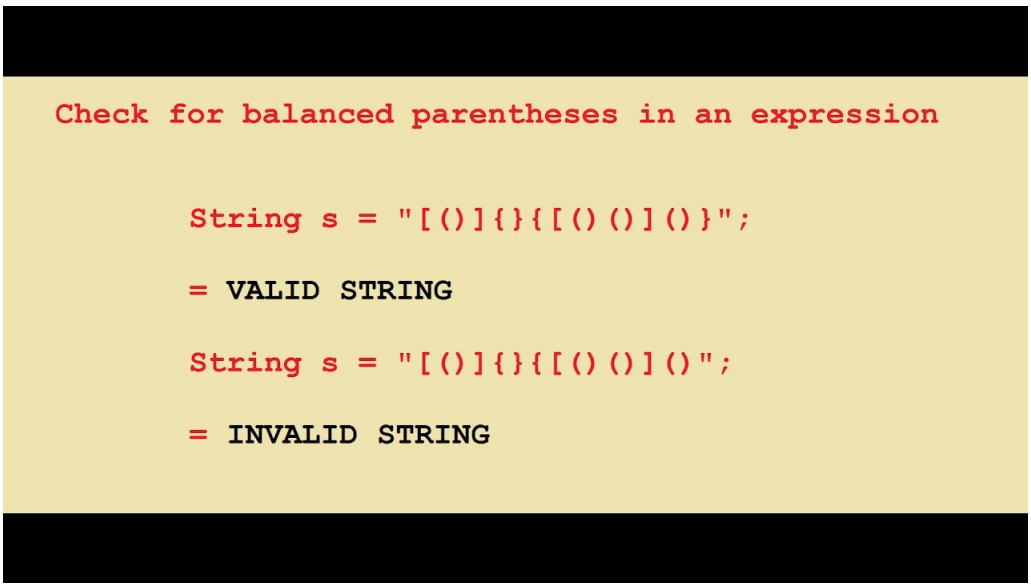
Entrada	Saída
`a * b - (2 + c)`	`Correta`

Entrada	Saída
`)3+b*(2-c)(`	`Incorreta`

Entrada	Saída
`(a+b*(2-c)-2+a)*2(`	`Correta`

Observação: Perceba que a primeira letra, e somente ela, das palavras *correta* e *incorreta* está grafada em maiúscula. Você deve fazer com que seu programa grafe exatamente desta maneira.

Entrada	Saída
`2*(3-a))`	`Incorreta`



7 Expressões

 (++++)

Pedrinho e Zezinho estão precisando estudar a respeito da resolução de expressões matemáticas para uma prova que irão, em breve, prestar. Para isso, eles querem resolver muitos exercícios antes da prova. Como sabem programar, então decidiram elaborar um "gerador de expressões matemáticas" – GEM – em $\mathbb{C}$.
O gerador de expressões que eles criaram funciona em duas fases:

1ª fase – É gerada uma cadeia de caracteres c que contém apenas os caracteres {, [, (, },] e). Ou seja: abre chaves, abre colchetes, abre parênteses, fecha chaves, fecha colchetes e, por fim, fecha parênteses.

2ª fase – O gerador adiciona os números e operadores na estrutura criada na primeira fase.

Uma cadeia de caracteres c é dita "bem definida" (ou válida) se atende às seguintes propriedades:

1. $c = \lambda$, ou seja, c é uma cadeia de caracteres vazia (não contém nenhum caractere);
2. c é formada por uma cadeia "bem definida" envolvida por parênteses, colchetes ou chaves. Portanto, se a cadeia S é "bem definida", então as cadeias (S), $[S]$ e S também são bem definidas;
3. c é formada pela *concatenação* de duas cadeias "bem definidas". Logo, se as cadeias x e y são "bem definidas", a cadeia xy é "bem definida".

Depois que Pedrinho e Zezinho geraram algumas expressões matemáticas, eles perceberam que havia algum erro na primeira fase do GEM, pois algumas das cadeias geradas **não eram** do tipo "bem definida".
Eles querem começar a resolver as expressões o mais rápido possível, e sabendo que você é um ótimo programador, resolveram pedir sua ajuda: escreva um programa que "*dadas várias cadeias geradas na primeira fase, determine quais delas são bem definidas e quais não são*".

Entrada

A entrada é composta por diversas instâncias e, por isso, a primeira linha da entrada contém um inteiro t indicando o número de instâncias – $1 \leq t \leq 20$. Em seguida temos t linhas, cada uma com uma cadeia c. Sabe-se que o tamanho de c pode variar de 1 (um) a 100000 (cem mil) caracteres.

Saída

Para cada instância da entrada, imprima uma linha contendo a letra *S* se a cadeia é do tipo “bem definida”, ou a letra *N*, caso contrário.

Exemplos

Entrada	Saída
12 ( ) ([]) { } (] } { ([{ }]) { } [ ] ( ) ( )) { [ ] ((({ } { }()[[]])() { }) { } ((((((((((({ ([]) }]))))))))))	S S S N N S S N N N S N

Entrada	Saída
3 (()]]][[(())	N N S

8 PopPush – A Estação de Trem

 (+++)

Há uma famosa estação de trem na cidade *PopPush*. Esta cidade fica em um país de relevo incrivelmente acidentado e a sua estação foi criada no último século. Infelizmente os fundos financeiros eram extremamente limitados naquela época e, por isso, foi possível construir somente uma pista. Além disso, devido a problemas de espaço, foi feita uma pista apenas até a estação.
A tradição local é que todos os "comboios" que chegam vindo da direção A continuam na direção B com os vagões reorganizados, de alguma forma.
Suponha que o trem que está chegando da direção A tem n vagões numerados sempre em ordem crescente: $1, 2, 3, \ldots, n$. O primeiro vagão que chega é o de número 1 e o último que chega é o de número n. Existe um "Chefe de Reorganizações de Trens" que quer saber se é possível reorganizar os vagões para que os mesmos saiam na direção B na ordem $a_1, a_2, a_3, \ldots, a_n$.
O *chefe* pode utilizar qualquer estratégia para obter a saída desejada. Por exemplo: se o chefe quer saber se a saída 5,4,3,2,1 é possível em B, nesse caso, basta o *chefe* deixar todos os vagões entrarem na estação (do 1 ao 5) e, depois, retirar um a um: retira o 5, retira o 4, retira o 3, retira o 2 e por último retira o 1. Desta forma a resposta seria **Yes**. Vagão que entra na estação só pode sair para a direção B e é possível incluir quantos vagões forem necessários para retirar o primeiro vagão desejado.

Entrada

A primeira linha da entrada contém $n \in \mathbb{N}^*$, o número vagões, com $1 \leq n \leq 1000$.
Em cada uma das linhas seguintes – podem ser m linhas, com $1 \leq m \leq 10^6$, há uma permutação dos valores $1, 2, 3, \ldots, n$ que representa a ordem de saída desejada pelo *chefe*.

Saída

Para cada caso de teste fornecido, o programa deve imprimir, numa única linha, a mensagem **sim** se for possível obter a ordem de saída desejada, ou **não**, do contrário.

Exemplos

Entrada	Saída
5 5 4 3 2 1	sim

Entrada	Saída
5 1 2 3 4 5	sim

Entrada	Saída
5 4 5 1 2 3	nao

Observação: Note que a palavra *não* está escrita em letras minúsculas e sem acentuação gráfica na saída de teste. É assim que você deverá realizá-la.

Entrada	Saída
7 7 6 5 4 3 1 2	nao

Entrada	Saída
7 1 2 3 4 5 7 6	sim

9 Caminho de um Astronauta

 (++)

A família de Marcos Pontes, uma criança de nove anos, acaba de se mudar para uma nova cidade, e hoje é o seu primeiro dia de escola.
Como ele é um menino muito distraído, pois vive pensando em ser astronauta, sua mãe fez uma *lista de instruções* para que saiba, sem erro, andar de sua casa até a escola.
Cada instrução descreve uma curva que ele deve fazer. Por exemplo, a lista:

INSTRUÇÕES
D
QUINZE
D
F
D
ESCOLA

Ela significa que Marcos deve "virar à direita" (D) na rua QUINZE, em seguida deve "virar à direita" na rua F e, finalmente, "virar à direita" na ESCOLA.
Você foi incumbido de fornecer as instruções para que Marcos faça o caminho contrário ao que recebeu para chegar ao seu destino. Para isto você deve elaborar um programa de computador, escrito em $\mathbb{C}$, para realizar a tarefa de *geração* das instruções a serem fornecidas para Marcos.
Você pode assumir que a lista que Marcos recebe contém pelo menos duas e, no máximo, cinco instruções. Pode também pressupor que cada linha contém, no máximo, 10 caracteres que sempre são grafados utilizando letras maiúsculas e que a última instrução sempre é para "virar" para a ESCOLA.

Entrada

A primeira linha da entrada contém um número natural t, indicando o número de casos de teste. Sabe-se $1 \leq t \leq 100$.
Para cada caso de teste há uma sequência de pares de linhas com a lista de instruções para o caminho de casa até a ESCOLA. A primeira linha do par é uma instrução de virar à *direita* – D – ou à *esquerda* – E. A

segunda linha é o nome da rua, exceto na última linha de cada caso de teste que, sempre, conterá a palavra ESCOLA.

Saída

Seu programa deve imprimir, na saída padrão, e para cada caso de teste, uma sequência de linhas que especifiquem o caminho de volta da escola até a casa de João.

Exemplos

Entrada	Saída
2 D QUINZE D QUATRO D ESCOLA E PRINCIPAL D ESCOLA	Vire a ESQUERDA na rua QUATRO. Vire a ESQUERDA na rua QUINZE. Vire a ESQUERDA na sua CASA. Vire a ESQUERDA na rua PRINCIPAL. Vire a DIREITA na sua CASA.

Observação: Cada uma das instruções deve, obrigatoriamente, seguir ao padrão anteriormente exemplificado. Note que, apesar de necessária na Língua Portuguesa, a acentuação foi eliminada das orações. O correto seria, por exemplo, “Vire à ESQUERDA na rua QUATRO.”.

Entrada	Saída
3 E DEZ E OITO D CINCO D ESCOLA E DEZ E OITO E QUATRO D DOIS D ESCOLA D DEZ D OITO E QUATRO E ESCOLA	Vire a ESQUERDA na rua CINCO. Vire a ESQUERDA na rua OITO. Vire a DIREITA na rua DEZ. Vire a DIREITA na sua CASA. Vire a ESQUERDA na rua DOIS. Vire a ESQUERDA na rua QUATRO. Vire a DIREITA na rua OITO. Vire a DIREITA na rua DEZ. Vire a DIREITA na sua CASA. Vire a DIREITA na rua QUATRO. Vire a DIREITA na rua OITO. Vire a ESQUERDA na rua DEZ. Vire a ESQUERDA na sua CASA.

10 Notação Polonesa Inversa

 (++)

Usualmente uma expressão aritmética *e* é escrita com os operadores entre os respectivos operandos (no caso de operadores binários) e, por isso, essa notação é chamada de *Infixa*. Por exemplo:
$e = (2+4\cdot(6-1)/2)$.
Uma forma alternativa de escrever *e* é utilizando-se da *Notação Polonesa Inversa* (ou *Notação Posfixa*). Essa notação foi inventada pelo filósofo e cientista da computação australiano Charles Hamblin em meados dos anos 1950, com base na *Notação Polonesa*, introduzida em 1920 pelo matemático polonês Jan Lukasiewicz. A tabela a seguir mostra alguns exemplos de operações e notações nessas duas notações:

EXPRESSÃO	**NOTAÇÃO INFIXA**	**NOTAÇÃO POL. INVERSA**
a + b	a + b	a b +
$\frac{a+b}{c}$	(a + b) / c	a b + c /
$\frac{a\cdot b-c\cdot d}{e\cdot f}$	((a * b) - (c * d))/(e * f)	a b * c d * - e f * /

Note-se que a ordem dos operandos é a mesma nas notações infixa e posfixa. Além disso, a notação posfixa dispensa o uso de parênteses.

Tarefa

A sua tarefa é construir um programa de computador para *traduzir* expressões escritas com a notação infixa para a notação posfixa.
As expressões são formadas por constantes numéricas, variáveis e operadores aritméticos. Para simplificar, considere que na expressão infixa são usadas apenas as operações aritméticas de soma, subtração, divisão, multiplicação e exponenciação ($+$, $-$, $/$, $*$ e $\hat{}$). Considere também que nomes de variáveis são formados por apenas uma letra maiúscula (A, B, C, ..., Z), as constantes numéricas são formadas por apenas um dígito (1, 2, ..., 9) e não há espaços antes ou após um operador aritmético, seja no início ou no final da expressão.
Se para cada símbolo de "abre parênteses" em uma expressão na notação infixa há um correspondente símbolo de "fecha parênteses", diremos que a expressão está *bem formada*.

Entrada

Os dados de entrada são compostos por vários casos de teste. Cada caso de teste contém uma expressão aritmética escrita na notação infixa, com n operadores, sabe-se que $0 \leq n \leq 100$.

Saída

A saída, para cada caso de teste fornecido na entrada, consiste da expressão na notação infixa numa linha e, na linha seguinte, sua correspondente expressão aritmética escrita na notação posfixa ou, alternativamente, a mensagem *ERRO*, caso a expressão na notação infixa não seja *bem formada*. Imprima também uma linha em branco após cada caso de teste.

Exemplo

ENTRADA	SAÍDA
0 / 4	0/4
(E/5+(8)^U+4)+7^(9)∗2	0 4 /
5∗4/(M+8+T−B∗8^T^T	
Q∗H∗S/2∗(Q/(J)^Y∗C	(E/5+(8)^U+4)+7^(9)∗2
	e 5 / 8 U ^+ 4 + 7 9 ^2 * +
	5∗4/(M+8+T−B∗8^T^T
	ERRO
	Q∗H∗S/2∗(Q/(J)^Y∗C
	ERRO

11 A Lista de Arya (série "Game of Thrones")

(+)

Lista de Arya: "Cersei. Walder Frey. Montanha. Meryn Trant."

Para se manter motivada, Arya sempre lembra a lista de inimigos que ela mais odeia. O principal objetivo de sua jornada é acabar com todos na sua lista!

Entretanto, às vezes algum inimigo dela pode ser morto por outra pessoa. Quando ela descobre que tal inimigo morreu, ela o remove da sua lista. Além disso, Arya também pode fazer novos inimigos durante sua jornada. Quando ela faz um novo inimigo, tal inimigo é incluído na sua lista.

Arya quer acabar com seus inimigos um por um, na mesma ordem em que aparecem na sua lista. A qualquer momento, ela pode se perguntar quanto tempo irá levar para acabar com todos que estão entre dados dois inimigos. Para tal, dados dois inimigos, digamos I_a e I_b, ela deve determinar quantos inimigos estão na lista entre I_a e I_b, excluindo ambos.

Ajude Arya respondendo tais perguntas por meio da elaboração de um programa de computador.

Entrada

A primeira linha da entrada contém um número inteiro $n(1 \leq n \leq 1000)$ que corresponde ao número de inimigos inicialmente na lista de Arya. Considere que todas as pessoas são numeradas de 1 a 1000, inclusive, não havendo repetição de números.

A próxima linha contém n inteiros, descrevendo a lista inicial de Arya.

As linhas seguintes descrevem as operaçõesa serem realizadas na lista. Cada operação pode estar em um dos seguintes formatos:

- **I p e** $(1 \leq e, p \leq 100)$: Insira a pessoa p depois do inimigo e na lista. É garantido que e está na lista, e p não está na lista;
- **R e** $(1 \leq e \leq 100)$: Remova o inimigo e da lista. É garantido que e está na lista;
- **Q a b** $(1 \leq I_a, I_b \leq 100)$: Determine quantos inimigos estão na lista entre I_a e I_b, excluindo ambos. É garantido que I_a e I_b estão na lista;
- **F**: Termina aquela sequência de operações.

Saída

Imprima uma linha para cada operação do tipo **Q** com sua resposta.
Observação: Caso I_a e I_b sejam iguais, deve-se exibir como saída o valor -1 (menos um).

Exemplos

Entrada	Saída
3 3 8 2 Q 3 2 I 9 8 Q 3 2 R 8 I 1 2 Q 1 9 F	1 2 1

Entrada	Saída
3 3 8 2 Q 3 3 I 9 8 Q 3 8 R 8 I 1 2 Q 3 1 F	-1 0 2

Entrada	Saída
3 3 8 2 Q 2 2 I 9 8 Q 8 3 R 8 I 1 2 Q 1 3 F	1 0 2

12 Micalatéia

 (+)

A cidade de Pentescopéia é muito atrasada (ainda usam telefones da década de 1980!), e somente três pessoas possuiam telefone, uma delas era Micalatéia, irmã de Hermanoteu.
Micalatéia possui um *hobbie* muito esquisito: ela anota em sua agenda de contatos o número de vezes em que ela ligou para uma determinada pessoa. A agenda de Micalatéia registra, até o momento, o seguinte:

- Hermanoteu 4523-2248 300 ligações
- Oolonéia 4523-4887 299 ligações

Como ultimamente tem aumentado o número de pessoas com telefone em Pentescopéia, sua agenda está crescendo e ela pediu para que você criasse um programa, em $\mathbb{C}$, que a permitisse realizar as seguintes funções:

- Inserir um novo contato;
- Remover um contato;
- Registrar quem fez uma ligação.

E que essa lista seja ordenada segundo o número de ligações (ordem descrescente).

Entrada

Cada linha do arquivo de entrada pode possuir algum dos seguintes formatos:

- **I nome tel v**: Insira a pessoa com **nome** (de até 20 caracteres), e que possui o número de telefone expresso em **tel**, sendo **v** o número de vezes que ela ligou para esta pessoa;
- **R nome**: Remova a pessoa que possua o nome denotado por **nome** da lista;
- **L nome**: Aumente o número de ligações, **v**, que ela fez para a pessoa cujo nome é **nome**;
- **F**: Termine esta sequência de operações.

Observações: (1) É garantido que **nome** está na lista, e que não exista mais de uma pessoa com o mesmo nome; (2) Sempre que o número de ligações forem iguais, o primeiro que possuir esse número fica na frente; e (3) Não há pessoas com nome composto.

Saída

Imprima a lista de nomes, e seus respectivos telefones, do primeiro ao último elemento de acordo com a formatação apresentada nos exemplos a seguir.

Exemplos

Entrada	Saída
L Ooloneia I Dirineia 4523-6667 0 I Mirineu 4523-1313 1 L Dirineia L Dirineia F	Hermanoteu - 4523-2248 Ooloneia - 4523-4887 Dirineia - 4523-6667 Mirineu - 4523-1313

Entrada	Saída
L Hermanoteu I Dirineia 4523-6667 0 I Mirineu 4523-1313 2 L Dirinéia L Dirinéia F	Hermanoteu - 4523-2248 Ooloneia - 4523-4887 Mirineu - 4523-1313 Dirineia - 4523-6667

Entrada	Saída
R Hermanoteu I Dirineia 4523-6667 299 I Javeu 4523-4343 299 I Mirineu 4523-1313 299 R Javeu L Mirineu L Dirineia L Ooloneia F	Mirineu - 4523-1313 Dirineia - 4523-6667 Ooloneia - 4523-4887

13 Três Cortes

 (+)

Três cortes é um jogo muito popular nas escolas de Anápolis. O jogo consiste em, após o terceiro toque (com os pés) na bola, alguém "*corta*" e tenta "*carimbar* outra pessoa.

As pessoas estão em um círculo, quem é *carimbado* vai para o meio do círculo. Se uma pessoa der dois toques consecutivos na bola, também vai para o meio círculo. Se alguém do meio do círculo for carimbado, ele é salvo e pode voltar a brincadeira, posicionando-se, novamente, como uma das pessoas que está no círculo, inserindo-se em qualquer posição que deseje. Por fim, se alguém erra em qualquer ponto da jogada, o jogo recomeça.

Numa certa escola em que este jogo acontece frequentemente, foi definida uma regra especial: se uma pessoa que está no meio do círculo for acertada, ela deve ao retornar à brincadeira se posicionar ao lado direito de quem o *salvou*.

Tarefa

Você deverá, conforme as instruções a seguir, criar um programa $\mathbb{C}$ que seja capaz de simular esta brincadeira.

Entrada

As primeiras linhas da entrada contém os nomes de quem participará da brincadeira. Deve-se ler até que seja digitada a *string* FIM, com todas as letras maiúsculas, indicando o término dos nomes.

A próxima linha contém um número inteiro $n(1 \leq n \leq 1000)$ que representa o número de jogadas a serem realizadas.

As *n* linhas seguintes possuem os nomes de quem está executando os toques, o último nome é da pessoa que foi *carimbada*. No lugar de qualquer um desses quatros nomes pode vir a palavra **ERROU**, significando que aquela pessoa errou o que queria fazer, ou seja, dar um passe ou carimbar alguém.

Observação: É garantido que **nome** está na lista, e que não exista mais de uma pessoa com o mesmo nome, como também nenhum nome é composto.

Saída

Imprima quem estará no círculo após a realização das *n* jogadas programadas.

Exemplos

Entrada	Saída
Andreia Afonso Fernanda Joao Maria Pedro Walter FIM 5 Andreia Pedro Joao Afonso Joao Maria Pedro Andreia Joao Joao Pedro Walter ERROU Pedro Maria Fernanda ERROU	Fernanda Maria Pedro Walter

Entrada	Saída
Andreia Afonso Fernanda Joao Maria Pedro Walter FIM 5 Andreia Pedro Joao Afonso Joao Maria Pedro Afonso Joao Joao Maria Pedro Pedro ERROU	Andreia Fernanda Maria Afonso Walter

Entrada	Saída
Andreia Afonso Fernanda Joao Maria Pedro Walter FIM 5 ERROU Maria ERROU Fernanda Andreia ERROU Walter Fernanda Maria ERROU ERROU	Andreia Afonso Fernanda Joao Maria Pedro Walter

Entrada	Saída
Andreia Fernanda Joao Walter FIM 5 Andreia Fernanda ERROU ERROU Joao Joao Andreia Walter Andreia Fernanda Walter Andreia Walter ERROU	Andreia Fernanda Walter

14 Bobinho

 (+)

Em um treino da Seleção Brasileira de Futebol, alguns jogadores decidiram adicionar algumas regras na brincadeira conhecida por “bobinho”: (1) era permitido um número ilimitados de toques por um mesmo jogador. O bobinho teria que tomar a bola *duas vezes* para voltar à roda, e ele voltaria no início dela. Quem perde a bola vira “bobinho”. Se somente restar um jogador, ele vence aquela partida de “bobinho”.

Tarefa

Você deverá, conforme as instruções a seguir, criar um programa $\mathbb{C}$ que seja capaz de simular esta brincadeira.

Entrada

As primeiras linhas contém os nomes de quem participará da brincadeira, deve-se ler até que seja identificada a digitação da *string* FIM, sempre com todas as letras maiúsculas.

A linha seguinte contém o nome de quem começa como “bobinho”.

A próxima linha apresenta uma número inteiro $n(1 \leq n \leq 30)$, que representa o número de “roubadas” de bola. As n linhas seguintes possuem os nomes de quem perdeu a bola, seguido do nome de quem roubou a bola.

Observação: É garantido que os nomes estarão na lista, e que não exista mais de uma pessoa com o mesmo nome, como também não haverá nome composto.

Perceba que como esta é uma lista do tipo “não ordenada”, sempre insira antes do primeiro elemento, assim a inserção será $\mathcal{O}(1)$.

Saída

Imprima os nomes, um por linha, de quem está na roda após as n roubadas de bola. Se houver um vencedor imprima Vencedor: nome_do_vencedor.

Exemplos

Entrada	Saída
Marcelo Paulinho Willian Jesus Casemiro Coutinho Neymar FIM Casemiro 5 Willian Casemiro Paulinho Casemiro Marcelo Willian Jesus Willian Willian Paulinho	Casemiro Neymar Coutinho

Entrada	Saída
Marcelo Paulinho Willian Jesus Casemiro Coutinho Neymar FIM Neymar 5 Willian Neymar Paulinho Willian Marcelo Paulinho Jesus Marcelo Casemiro Jesus	Vencedor: Coutinho

Entrada	Saída
Marcelo Paulinho Willian Jesus Casemiro Coutinho Neymar FIM Neymar 5 Willian Neymar Paulinho Willian Marcelo Paulinho Jesus Marcelo Casemiro Neymar	Neymar Coutinho

15 Brincadeira

 (+)

Para acalmar seus cinco netos, Dona Arlete propôs uma brincadeira:

Um neto era *vendado*, e o restante se agrupava em um círculo, sendo que a vovó iniciava esse círculo. O neto *vendado* dizia um nome (de outro dos netos) e uma direção: direita ou esquerda.

Se em até dois passos, sem contar a vovó, ele acertasse o nome de um dos outros netos, então marcava um ponto e o primo (ou irmão) cujo nome foi dito deixava o círculo. Esse processo se repetia para cada neto presente no círculo. Não se podia repetir os nomes.

É a vez de Paulo ficar vendado, dado uma disposição dos outros primos no círculo, e os nomes e direções ditas por Paulo, determine sua pontuação.

Entrada

As primeiras linhas contém os nomes dos netos, do primeiro do círculo ao último, sendo que a leitura de nomes deve ser encerrada pela digitação da palavra FIM, com todas as letras maiúsculas.

Na sequência, há uma linha por neto no círculo, onde está o nome de um neto e a direção (esquerda ou direita).

Observação: Lembre-se que a Dona Arlete, a vovó, é que inicia o círculo.

Saída

Imprima a pontuação de Paulo.

Exemplo

Entrada	Saída
Ana Joaquim Henrique Marcela Carlos Sabrina Loys FIM Henrique direita Ana direita Joaquim direita Marcela direita Carlos esquerda Sabrina esquerda Loys esquerda	5