Sumário

1 Fila do SUS

Os pacientes que chegam à fila do *Sistema Único de Saúde* (SUS) passam por uma *triagem* e, imediatamente, vão para a fila de atendimento [1].

No processo de triagem, um(a) enfermeiro(a) anota o horário de entrada do(a) paciente e quantos minutos ele(a) tem até que sua condição de saúde se torne crítica, ou seja, momento a partir do qual sua morte será inevitável. Uma característica *especial* é que o(a) enfermeiro(a) **nunca erra em seu diagnóstico** e, portanto, consegue predizer, de maneira inequívoca, em quanto tempo o paciente atingirá este estado crítico. Além disso faz esta atividade de maneira instantânea, ou seja, a identificação do estado de um(a) paciente não consome nenhum tempo.

Sabe-se também que os(as) pacientes são atendidos(as) de 30 em 30 minutos, com pontualidade britânica e com atendimentos sempre sendo iniciados em *horas cheias* ou *meias-horas*: 07h, 07h30min, 08h e, assim, sucessivamente. O início da triagem e do atendimento se dá, pontualmente, às 07h de cada dia.

Se não há nenhum(a) paciente sendo atendido(a) e a fila está vazia – Isto ocorre? Eu não acredito – o(a) primeiro(a) paciente é atendido(a) no instante que chega à triagem. O(A) médico(a) sempre atende até o(a) último(a) paciente na fila.

A preocupação é se algum(a) paciente atingiu a sua *condição crítica* enquanto está na fila de atendimento, ou seja, ainda não tenha sido iniciado seu atendimento.

Diante deste cenário, você foi convidado(a) a implementar um programa de computador, em C, que seja capaz de verificar na fila quantos pacientes atingem a sua condição crítica.

Entrada

A primeira linha da entrada contém um número inteiro n, $0 < n \leq 50$, o número de pacientes que chegaram à triagem.

As n linhas seguintes possuem, cada uma, uma tríade de valores inteiros maiores do que zero: *h, m, c,* sendo que $7 < h < 19$ e $0 \leq m < 60$ e $0 \leq c \leq 720$. Os valores de *h* e *m* correspondem, respectivamente, à hora e minuto em que o(a) paciente chega à triagem. O(A) paciente da linha *i* sempre chega antes de, e no máximo junto com, o(a) paciente da linha $(i + 1)$. O valor *c* é o tempo, expresso em minutos, antes daquele(a) paciente atingir a condição crítica em seu estado de saúde.

Saída

O programa deverá imprimir o número de pacientes que atingiram a condição crítica ainda na fila para atendimento.

Exemplos

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

[1] Lembre-se: Isto é apenas um exercício de programação e, portanto, não há nenhuma fidedignidade com os acontecimentos reais no SUS do Brasil

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

2 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 e, portanto, nunca existe um espaço vago entre duas pessoas consecutivas na fila. A fila inicialmente contém n pessoas, cada uma com um identificador diferente: m_i. Estes identificadores são, vulgarmente, chamados de "*senhas*".

Pedro, 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* da fila e a sequência de identificadores das pessoas que deixaram a fila, na ordem em que elas a abandonaram. Ele também sabe que, após o *estado inicial*, nenhuma pessoa entrou mais na fila.

Ele deseja saber qual será o *estado final* desta fila, mas não quer fazer, manualmente, a simulação das saídas das pessoas. Pedro 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, C, que resolva este problema.

Entrada

A primeira linha contém um número natural n representando a quantidade de pessoas inicialmente na fila , sendo que $1 \leq n \leq 60000$.

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

A terceira linha contém um natural s, $1 \leq s \leq n$, que representa a quantidade de pessoas que deixaram a fila.

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

Saída

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

Exemplos

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

3 Branca de neve e os *n* anões

Branca de Neve e os *n* anões 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 princesa tecnológica, que acessa e atualiza diariamente a sua rede social favorita fazendo o *upload* das fotos que considera bonitas.
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. 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. 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. 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 quer verificar, para um conjunto de *m* fotos, quantas são *fotos bonitas* e qual a cor predominante nas toucas. Como você não quer fazer isto manualmente, resolveu escrever um programa de computador, em C, para cumprir a tarefa.

Entrada

A primeira linha da entrada conterá dois números naturais *n* e *c* que representam respectivamente o número de anões e o número de cores diferentes de toucas, sendo que $3 \leq n \leq 1000$ e $1 \leq c \leq n$. Cada cor de touca é representado por um número inteiro *i* onde que $1 \leq i \leq c$.

A segunda linha da entrada conterá a fila dos *n* anões contendo as cores das suas respectivas toucas.

A terceira linha da entrada conterá o número inteiro m que representa o número de fotos tiradas por Branca de Neve ($1 \leq m \leq 1000$)

As *m* linhas seguintes conterão a configuração de cada uma das *m* fotos tiradas. A configuração é representada por dois números i e j que indicam quais anões apareceram na foto ($1 \leq i \leq j$). O número *i* indica que o primeiro anão que aparece na foto estava na *i-ésima* posição da fila e o número *j* indica que o útimo anão que aparece na foto estava na *j-ésima* posição da fila. Lembre-se que todos os anões que estão entre a posição i e a posição j também aparecem na foto.

Saída

A saída deverá conter, por caso 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 "bonita X", onde X é a cor predominante nas toucas dos anões naquela foto. Lembre-se que, em nosso caso, a cor é representada por um número natural. Do contrário, a mensagem deve ser "feia".

Exemplos

Entrada	Saída
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	feia bonita 1 feia bonita 1 feia bonita 2 feia bonita 3

Entrada	Saída
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	feia feia feia feia feia feia

4 Balanceamento de parênteses

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

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

é uma expressão correta, pois há um "abre" parêntese e um "fecha" parêntese.
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 parêntese que "fecha" deve ter um outro parêntese que "abre" correspondente, bem como não pode haver parêntese 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, em C, que seja capaz de verificar se uma dada expressão *e*, fornecida como entrada, está correta no que diz respeito à parentetização.

Observação: É obrigatório que seu programa utilize uma estrutura de dados do tipo "*dinâmica*" para implementar a solução para este problema.

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, ou seja, desconsideram-se todos os demais elementos (numerais, espaços em branco e símbolos de operação aritmética).

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 todas as letras das palavras *correta* e *incorreta* estão grafadas em minúscula. Você deve fazer com que seu programa grafe *exatamente* desta maneira.

5 Caminho

A família de Luna, uma menina de dez anos, acaba de se mudar para uma nova cidade, e hoje é o seu primeiro dia de aulas na sua "nova" escola.
Como ela é uma menina muito distraída, sua mãe fez uma *lista de instruções* para que ela saiba, sem erro, andar de sua casa até a escola.
Cada instrução descreve uma curva que ela deve fazer para contornar uma esquina da cidade.
Por exemplo, a lista:

INSTRUÇÕES
DIREITA
QUINZE
DIREITA
FRAMBOESA
DIREITA
ESCOLA

Ela significa que Luna deve "virar à direita" na rua QUINZE, em seguida deve "virar à direita" na rua FRAMBOESA e, finalmente, "virar à direita" para chegar à ESCOLA.
Você foi incumbido de fornecer as instruções para que Luna faça o caminho contrário ao que recebeu para chegar ao seu destino. Para isto você deve elaborar um programa de computador, escrito em C, para realizar a tarefa de *geração* das instruções a serem fornecidas para Luna em seu retorno para casa.
Você pode assumir que a lista que Luna recebe contém pelo menos duas e, no máximo, dez instruções. Pode também pressupor que cada linha contém, no máximo, 20 caracteres que sempre são grafados utilizando letras maiúsculas e que a última instrução sempre é para "virar" para a ESCOLA.

Entrada

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* – DIREITA – ou para vira à *esquerda* – ESQUERDA. 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 Luna.

Observação: Cada uma das instruções deve, obrigatoriamente, seguir ao padrão contido nos exemplos. Note que, apesar de necessária na Língua Portuguesa, a acentuação foi eliminada das frases. O correto seria, por exemplo, "Vire à ESQUERDA na rua QUATRO.".
Perceba também que todas as frases terminam com "." (ponto final).
A saída de seu programa deve, portanto, ser *exatamente* igual à apresentada em cada caso de teste.

Exemplos

Entrada	Saída
DIREITA QUINZE DIREITA QUATRO DIREITA ESCOLA	Vire a ESQUERDA na rua QUATRO. Vire a ESQUERDA na rua QUINZE. Vire a ESQUERDA para chegar em CASA.

Entrada	Saída
ESQUERDA PRINCIPAL DIREITA ESCOLA	Vire a ESQUERDA na rua PRINCIPAL. Vire a DIREITA para chegar em CASA.

Entrada	Saída
ESQUERDA R6 ESQUERDA R4 ESQUERDA R9 DIREITA R7 ESQUERDA ESCOLA	Vire a DIREITA na rua R7. Vire a ESQUERDA na rua R9. Vire a DIREITA na rua R4. Vire a DIREITA na rua R6. Vire a DIREITA para chegar em CASA.

6 Notação polonesa inversa

Usualmente uma expressão aritmética *e* é escrita com os operadores binários entre os respectivos operandos e, por isso, essa notação é chamada de *infixa*.
Por exemplo:

$$e = (2 + 4 \cdot (6 - 1)/2)$$

ou, equivalentemente:

$$e = 2 + 4 \cdot \frac{(6-1)}{2}$$

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 Leonard Hamblin (1922-1985) em meados dos anos 1950, com base na *Notação Polonesa*, introduzida em 1920 pelo matemático polonês Jan Lukasiewicz (1878-1956).
A tabela a seguir mostra alguns exemplos de operações e suas notações:

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

Note 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 (+ , - , / , *, ^).
Considere também que nomes de variáveis são formados por apenas uma única letra maiúscula maiúscula (A, B, C, ..., Z), as constantes numéricas são formadas por apenas um dígito (1, 2, ..., 9) e há um único espaço em branco, antes e após, um operador aritmético.
Se para cada símbolo de "abre parêntese" em uma expressão na notação infixa há um correspondente símbolo de "fecha parêntese", diremos que a expressão está *bem-formada*. Do contrário ela é considerada *mal-formada*.

Entrada

Os dados de entrada são compostos por vários casos de teste, sendo que a primeira linha contém um único número natural t que indica o número de casos de teste, com $1 \leq t \leq 10$.
Cada caso de teste está numa linha da entrada e contém uma expressão aritmética escrita na notação infixa, com n operadores, sabe-se que $0 \leq n \leq 100$. Cada símbolo (numeral, variável, abre parêntese, fecha parên-

tese ou operador) é sempre precedido e sucedido por um único espaço em branco, exceto quando ele é o primeiro ou o último da expressão.

Saída

A saída, de cada caso de teste fornecido, deve ser apresentada numa linha e consiste da correspondente expressão aritmética escrita na notação posfixa ou, alternativamente, a mensagem "mal-formada", caso a expressão na notação infixa seja *mal-formada*.
Da mesma forma que as entradas, cada linha da saída possui cada os seus símbolos (numeral, variável ou operador) precedido e sucedido por um único espaço em branco, exceto quando ele é o primeiro ou o último da expressão.

Exemplos

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

ENTRADA	SAÍDA
4 (6 + 4) * 2 (6 ^ 2 - 2 ^ 2) * 7 ((4 + 2) *(5 - 4)) ^ 2 7 * 6 + 5 * 4	6 4 + 2 * 6 2 ^ 2 2 ^ - 7 * 4 2 + 5 4 - * 2 ^ 7 6 * 5 4 * +

Atenção: Nos exemplos de casos de testes apresentados pode não parecer que há um ÚNICO ESPAÇO EM BRANCO entre cada um dos símbolos impressos, mas considere que HÁ.

7 A lista de Arya (série "Game of Thrones")

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 – o que é lamentável. 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 *a* e *b*, ela deve determinar quantos inimigos estão na lista entre *a* e *b*, excluindo ambos. Ajude Arya respondendo tais perguntas.

Entrada

A primeira linha contém um número inteiro *n* ($1 \leq n \leq 1000$) que corresponde ao número de pessoas inicialmente na lista de inimigos de Arya. Considere que todas as pessoas que Arya conhece são numeradas de 1 a 1000, inclusive extremos.
A próxima linha contém *n* inteiros, indicando as pessoas que estão na lista inicial de inimigos de Arya. As linhas seguintes descrevem as operações que podem ser realizadas na lista de Arya. Cada operação pode estar em um dos seguintes formatos:

`I p e`

Insira a pessoa de número *p* depois do inimigo de número *e* na lista. É garantido que *e* está na lista, e que *p* não está na lista, com $(1 \leq e, p \leq 1000)$. Deve imprimir, como saída, a expressão: "inserido" *p*.

`R p`

Remova o inimigo de número *p* da lista. É garantido que *p* está na lista. Deve imprimir, como saída, a expressão: "removido " *p*.

`C a b`

Determine a quantidade de inimigos estão na lista entre os inimigos de números *a* e *b*, excluindo ambos. É garantido que *a* e *b* estão na lista, mas não se sabe a ordem em que eles estão. Deve imprimir, como saída, a expressão: "quantidade " *x*, onde *x* é o número de inimigos.

`M`

Mostre os números os inimigos que estão na lista, a partir do primeiro em direção ao último. Deve imprimir, como saída, a expressão: "lista " $p_1\ p_2\ p_3\ ...\ p_n$, considerando que há *n* inimigos na lista de Arya. Se a lista estiver vazia, deve-se mostrar a mensagem "lista vazia".

`F`

Termina as operações com a lista de Arya e, por consequência, a execução do programa, imprimindo a palavra "fim".

Saída

Imprima uma linha por operação realizada, conforme mostram os exemplos a seguir.

Exemplos

Entrada	Saída
3 3 8 2 C 3 2 I 9 8 C 3 2 R 8 I 1 2 C 1 9 F	quantidade 1 inserido 9 quantidade 2 removido 8 inserido 1 quantidade 1 fim

Entrada	Saída
10 7 5 1 10 2 4 6 9 3 8 I 11 10 I 14 7 I 12 9 R 3 R 6 R 2 C 7 5 C 11 12 C 10 8 C 7 8 C 1 5 M F	inserido 11 inserido 14 inserido 12 removido 3 removido 6 removido 2 quantidade 1 quantidade 2 qunatidade 4 quantidade 8 quantidade 0 lista 7 14 5 1 10 11 4 9 12 8 fim

Entrada	Saída
10 7 5 1 10 2 4 6 9 3 8 C 7 5 C 10 3 C 9 8 C 1 8 F	quantidade 0 quantidade 4 quantidade 1 quantidade 6 fim

Entrada	Saída
2 1 10 I 9 1 I 7 9 I 8 1 C 1 10 C 8 7 C 9 7 R 1 R 10 C 8 7 F	inserido 9 inserido 7 inserido 8 quantidade 3 quantidade 1 quantidade 0 removido 1 removido 10 quantidade 1 fim

8 Micalateia

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

- Hermanoteu 4523-2248 300 ligações
- Ooloneia 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 C, que a permitisse realizar as seguintes funções:

- Inserir um novo contato;
- Remover um contato existente na lista;
- Registrar quem, da lista de contatos existente, fez uma ligação;
- E que essa lista seja ordenada por quem ela ligou mais vezes.

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), telefone `tel`, e `v`, correspondendo ao número de vezes que ela (Micalateia) ligou para esta pessoa;

`R nome`
Remova a pessoa que possui nome igual a `nome` da lista;

`L nome`
Aumenta, em uma unidade, o número de ligações que Micalateia fez para a pessoa cujo nome é `nome`;

`F`
Termina aquela sessão de operações na agenda.

Observações:

- É garantido que nome está na lista quando ocorre uma operação de remoção, como também não se insere alguém que já esteja na lista;
- Na lista nunca haverá duas pessoas com nomes idênticos, como também não haverá ninguém com nome composto;
- Sempre que o número de ligações for igual, o primeiro que possuir esse número fica na frente na lista.

Saída

A listagem de saída deverá conter uma pessoa por linha, na ordem em que estão dispostas a partir do início da lista na agenda, obedecendo ao seguinte formato:

- nome da pessoa;
- símbolo de hífen, precedido e sucedido por um único espaço em branco;
- número de telefone no formato xxxx-xxxx, onde cada "x" representa um dígito;
- número de ligações realizadas, precedidas por um único espaço em branco.

Exemplos

Entrada	Saída
L Ooloneia I Dirineia 4523-6667 0 I Mirineu 4523-1313 1 L Dirineia L Dirineia F	Hermanoteu - 4523-2248 300 Ooloneia - 4523-4887 300 Dirineia - 4523-6667 2 Mirineu - 4523-1313 1

Entrada	Saída
I Casimiro 7777-7777 10 I Zelia 8888-8888 100 I Machado 9999-9999 63 I Bilac 6666-6666 89 L Bilac L Bilac L Machado L Zelia L Casimiro F	Hermanoteu - 4523-2248 300 Ooloneia - 4523-4887 299 Zelia - 8888-8888 101 Bilac - 6666-6666 91 Machado - 9999-9999 64 Casimiro - 7777-7777 11

9 Brincadeira 1

Para acalmar seus cinco netos, Dona Arlete propôs uma brincadeira:
Um neto era *vendado* e o restante se agrupava formando um círculo, sendo que a vovó ocupava a posição de número "um" nesse círculo. O neto *vendado* dizia um nome (de outro dos netos) e uma direção: horário ou anti-horário.
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.
A vovó pediu seu auxílio: é a vez de Paulo ficar vendado e, dada a disposição inicial dos outros netos no círculo, bem como os nomes e direções ditas por Paulo, determine a pontuação feita por ele.
Você deverá escrever um programa C para realizar esta tarefa, segundo os padrões estabelecidos a seguir.

Entrada

As primeiras linhas contém, cada uma, o nome de um dos netos de dona Arlete, do primeiro do círculoao último, inseridos sempre no sentido horário. 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 (horário ou anti-horário).

Observação: Lembre-se que a Dona Arlete, a vovó, é que inicia o círculo ocupando sempre a posição de número "UM" e que, na entrada, as palavras "horário" e "anti-horário" são escritas sem a acentuação gráfica.

Saída

Imprima a pontuação de Paulo.

Exemplos

Entrada	Saída
Ana Joaquim Henrique Marcela Carlos Sabrina Loys FIM Henrique horario Ana horario Joaquim horario Marcela horario Carlos anti-horario Sabrina anti-horario Loys anti-horario	5

Entrada	Saída
Ana Joaquim Henrique Marcela Carlos Sabrina Loys FIM Loys anti-horario Sabrina anti-horario Henrique horario Marcela horario Ana horario Joaquim horario Carlos anti-horario	5

10 Brincadeira 2

Devido a uma forte e repentina chuva, ocorrida no primeiro dia de um acampamento infantil, as atividades recreativas ficaram limitadas e as crianças foram levadas para um ginásio de esportes existente no local.
No ginásio foi realizada uma *gincana* e uma das atividades consistiu em *agrupar* as crianças formando um círculo organizado no sentido anti-horário. Deste círculo as crianças são retiradas, uma a uma, até que restasse apenas uma criança, que se torna a *vencedora* daquela rodada da brincadeira.
No momento em que entra no círculo, cada criança recebe uma pequena ficha que contém um valor inteiro entre 1 a 500, incluindo os valores extremos. 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. Neste momento, a contagem reinicia na 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 é *ímpar*, a contagem é feita no sentido anti-horário. É importante destacar que a contagem sempre começa na criança que está ao lado da primeira criança que entrou e, por consequência, o sentido de contagem muda se o número for *par* ou *ímpar*. Para *par*, a imediatamente ao lado no sentido horário, e para *ímpar*, a imediatamente ao lado no sentido anti-horário.
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 durante o acampamento e, assim, o administrador pediu para que você, que é da equipe de Tecnologia da Informação do acampamento, desenvolva um programa de computador,em C, para que no próximo evento ele saiba, previamente, qual criança será vencedora de cada brincadeiracom base nas informações fornecidas pela entrada descrita a seguir.

Entrada

A primeira linha de entrada contém um número inteiro n, com $1 \leq n \leq 100$, indicando a quantidade de crianças que farão parte de cada círculo e, portanto, participarão da brincadeira.
Em seguida, as n linhas seguintes conterão duas informações: o nome e o valor ($1 \leq valor \leq 500$), que consta na ficha de cada criança, separados por um único espaço em branco entre eles e 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 (nomes compostos).

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

Entrada	Saída
5 Ada_Lovelace 100 Alan_Turing 91 John_Newmann 95 Carow_Shaw 20 Frances_Allen 81	Frances_Allen