



Observações:

- Uma máquina por equipe e não é permitido pegar material emprestado durante a maratona.
- Dispositivos eletrônicos não podem ser utilizados e devem ser guardados.
- O acesso à Internet é proibido até o final da maratona, exceto ao IP do BOCA.
- Este caderno de problemas contém 6 problemas e as páginas estão numeradas de 1 a 6.
- IP do Servidor: <http://200.17.76.19/boca/>

Equipe:			
Matrícula 1:		Nome 1:	
Matrícula 2:		Nome 2:	
Matrícula 3:		Nome 3:	

Problema a – Moedas

Arquivo-fonte: moedas.c | cpp | java)

O senhor José, possui uma vendinha de artesanato a muito tempo na cidade, todos os moradores e principalmente muitos turistas, já frequentaram a vendinha várias vezes comprando os mais variados produtos. Com o passar dos anos vendendo os seus produtos, ele acumulou uma quantidade imensa de moedas, de 1, 5, 10, 25, 50 centavos e de 1 real (representada por 100 centavos). Seu neto, recém-aluno de Sistemas de Informação, vendo aquela quantidade de moedas, começou a pensar de quantas maneiras possíveis o seu avô poderia devolver um mesmo troco aos seus clientes usando algumas das variadas moedas. Por exemplo, se José tivesse que devolver o troco de 20 centavos (0.2 reais) usando apenas moedas de 1, 5 ou 10 centavos, ele teria 9 maneiras de fazê-lo:

- 1) 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1
- 2) 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 5
- 3) 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 5 + 5 4) 1 + 1 + 1 + 1 + 1 + 5 + 5 + 5
- 5) 5 + 5 + 5 + 5 6) 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 10
- 7) 1 + 1 + 1 + 1 + 1 + 5 + 10 8) 5 + 5 + 10 9) 10 + 10

Dessa forma seu neto percebeu que dava para criar um programa para solucionar tal questionamento e deixou essa tarefa para você.

Entrada

A entrada consiste de dois números N e T que representam respectivamente, o número de moedas (um valor inteiro onde $1 \leq N \leq 6$) e o valor do troco, um número real com precisão máxima de duas casas, que representa o troco em reais. Em seguida, a entrada contém N números inteiros indicando quais as moedas distintas disponíveis para o troco que podem até as 6 variedades listadas anteriormente.








Saída

A saída consiste de um inteiro simples indicando a quantidade de maneiras que é possível entregar o troco com o tal conjunto de moedas

Exemplo de Entrada	Exemplo de Saída
3 0.2 1 5 10	9

Problema b**Jingle**Arquivo-fonte: *jingle*.(c | cpp | java)

Na música, uma nota tem um tom (sua frequência, que indica quão alto ou baixo é o som) e uma duração (por quanto tempo a nota é tocada). Neste problema estamos interessados apenas na duração das notas. Um jingle é dividido em uma sequência de compassos, e cada compasso é formado por uma série de notas. A duração de uma nota é indicada pelo seu formato. Usaremos letras maiúsculas para indicar a duração de uma nota. A seguinte tabela mostra as notas disponíveis:

Notas							
Identificador	W	H	Q	E	S	T	X
Duração	1	1/2	1/4	1/8	1/16	1/32	1/64

A duração de um compasso é a soma da duração de suas notas. Nos *jingles* do Gilberto, cada compasso tinha a mesma duração. Como Gilberto é iniciante, seu famoso professor *Johann Sebastian* III lhe ensinou que a duração de todo compasso deve ser sempre 1. Por exemplo, Gilberto escreveu a seguinte composição contendo cinco compassos, dos quais os quatro primeiros têm duração correta, mas o último está errado. Na representação da composição cada compasso é cercado por barras, e cada nota é uma das letras da tabela acima.

/HH/QQQQ/XXXTXTEQH/W/HW/

Gilberto gosta de computadores tanto quanto de música, e por isso quer um programa que determina, para cada uma de suas composições, quantos compassos têm a duração correta.

Entrada

A entrada contém vários casos de teste. Cada um é descrito numa única linha por uma *string* de tamanho entre 3 e 200 caracteres, inclusive, representando a composição. A composição começa e termina com barra '/'. Compassos na composição são separados por barra '/'. Cada nota de um compasso é representada por uma letra maiúscula, como descrito anteriormente. Você pode assumir que cada composição tem, pelo menos, um compasso, e que cada compasso tem, pelo menos, uma nota. Todos os caracteres da entrada são ou barra ou uma das sete letras maiúsculas usadas para representar as notas. O último caso de teste é seguido por uma linha contendo um asterisco.

Saída

Para cada caso de teste seu programa deve escrever em uma linha a mensagem "Composicao #:", onde # indica o número da composição iniciando em 1, um espaço em branco e em seguida um número inteiro, que é o número de compassos que tem duração correta na composição. Toda linha termina com um salto de linha.

Exemplo de Entrada	Exemplo de Saída
/HH/QQQQ/XXXTXTEQH/W/HW/ /W/W/SQHES/ /WE/TEX/THES/ *	Composicao 1: 4 Composicao 2: 3 Composicao 3: 0

Problema c**Swap**Arquivo-fonte: *swap.c* | cpp | java)

Você é o administrador de um cluster de computadores com discos que utilizam diversos sistemas de arquivos diferentes para armazenar os dados (*FAT*, *FAT32*, *NTFS*, *ext4*, etc.). Recentemente foi solicitado que houvesse apenas um tipo de sistema de arquivos. Este é um tremendo desafio, pois a mudança de um sistema de arquivos para outro pode causar um aumento na capacidade de armazenamento do disco ou uma diminuição.

Considerando que os discos estão completamente cheios de dados importantes e você não pode perder nenhum destes dados, para tornar a tarefa possível você terá que comprar um disco extra. Obviamente, você deseja economizar dinheiro e para isso o tamanho do novo disco deve ser o menor possível.

Você pode reformatar os discos em qualquer ordem. Antes de reformatar um disco, você deve mover todos os dados desse disco para um ou mais discos já reformatados. Considere que o disco extra já está com o sistema de arquivos correto e os dados podem ser divididos se necessário (ou seja, não é necessário colocar todos os dados no mesmo disco de origem). Na verdade, isso pode até ser impossível se alguns discos tiverem capacidade menor com o novo sistema de arquivos. Também é permitido que alguns dados acabem no disco extra.

Por exemplo, suponha que você tenha quatro discos A, B, C e D com capacidades de 6, 1, 3 e 3 GB, respectivamente. Após formatar para o novo sistema de arquivos, as capacidades se tornam 6, 7, 5 e 5 GB, respectivamente. Se você comprar um disco de apenas 1 GB de espaço extra, você pode mover os dados do disco B e, em seguida, reformatar o disco B. Agora você tem 7 GB livre no disco B, para que você possa mover os 6 GB do disco A e reformatá-lo. Finalmente, mova os 6 GB totais dos discos C e D para o disco A e reformate-os.

Entrada

A entrada começa com uma linha contendo um inteiro n ($1 \leq n \leq 10^6$), que é o número de discos em seu *cluster*. Logo em seguida estão as n linhas, cada uma descrevendo um disco como dois inteiros a e b , onde a é a capacidade com o antigo sistema de arquivos e b é a capacidade com o novo sistema de arquivos. Todas as capacidades são dadas em GB e satisfazem $1 \leq a, b \leq 10^9$.

Saída

Na saída, será impresso uma linha contendo um número inteiro que representa o total de capacidade extra em GB que é necessário comprar para reformatar os discos atuais.

Exemplo de Entrada	Exemplo de Saída
4 6 6 1 7 3 5 3 5	1
4 2 2 3 3 5 1 5 10	5

Problema d DVDs

Arquivo-fonte: dvds.(c | cpp | java)

Meu entusiasmo com fotografia me trouxe um pequeno problema: frequentemente meu HD fica lotado de fotos e pequenos vídeos! Decidi passar boa parte desses arquivos para DVDs.

Cada DVD tem sua capinha de plástico, e na frente dela escrevi informações sobre o conteúdo do DVD. Como a quantidade de DVDs aumentou bastante, comprei uma prateleira, onde os DVDs são colocados verticalmente.

Preciso da sua ajuda! Fiz um arquivo-texto com o título de todos os DVDs mas preciso que eles apareçam verticalmente, para poder ler os títulos mais facilmente quando colocar os DVDs na prateleira.

Quero um programa que escreva os títulos verticalmente. Preciso também de linhas entre cada título para, depois de imprimir, separar os títulos cortando o papel em cima das linhas.

Já fiz as medidas, e posso colocar 36 caracteres no espaço disponível da capinha, então todos os títulos devem ter no máximo 36 caracteres, com espaços em branco no final, se necessário. Se por acaso errei no tamanho de algum título e o fiz muito grande, imprima apenas os primeiros 36 caracteres.

Entrada

Haverá no máximo 50 títulos na entrada, um por linha, contendo de 1 a 100 caracteres quaisquer. Uma linha contendo apenas '#' indica o fim da entrada.

Saída

A saída deve conter os mesmos títulos da entrada, escritos verticalmente, na mesma ordem, da esquerda para direita. Deve haver uma coluna de '|' em cada lado da saída, e também separando os títulos. Deve haver ainda uma linha de '-' (caractere “menos”) no começo e no fim da saída, 1 por coluna. Toda linha termina com um salto de linha.

Exemplo de Entrada

```
012345678901234567890123456789012345
David and Jane's wedding, March 2002, Alexandria
Bahamas Holiday August 2001
#
```

Exemplo de Saída

```
-----
|0|D|B|
|1|a|a|
|2|v|h|
|3|i|a|
|4|d|m|
|5| |a|
|6|a|s|
|7|n| |
|8|d|H|
|9| |o|
|0|J|L|
|1|a|i|
|2|n|d|
|3|e|a|
|4|'|y|
|5|s| |
|6| |A|
|7|w|u|
|8|e|g|
|9|d|u|
|0|d|s|
|1|i|t|
|2|n| |
|3|g|2|
|4|,|0|
|5| |0|
|6|M|1|
|7|a| |
|8|r| |
|9|c| |
|0|h| |
|1| | |
|2|2| |
|3|0| |
|4|0| |
|5|2| |
-----
```

Problema e Estante de Livros

Arquivo-fonte: estante.(c | cpp | java)

Maria sempre adora ler livros e constantemente está comprando novos livros e tentando encontrar espaço para incluí-los em sua estante. No seu quarto ela tem uma estante apenas para os livros “a serem lidos”, os quais ela ainda não começou a ler.

Quando ela decide ler um desses livros, ela o remove da estante, deixando espaço para mais livros. Entretanto, às vezes ela compra um novo livro e o coloca na estante, mas por causa do espaço limitado, isto empurra um ou mais livros para fora da estante. Ela sempre coloca novos livros do lado esquerdo, o que pode fazer os livros mais à direita caírem da estante pela borda da direita. Ela pode remover um livro de qualquer lugar da estante quando quiser lê-lo.

Sua tarefa é escrever um simulador para acompanhar os livros adicionados e removidos da estante. No final da simulação, mostre os livros que permanecem na estante, na ordem da esquerda para a direita.

Os livros em cada simulação serão identificados por um ID que é um número inteiro, $0 < ID \leq 100$. Há três tipos de eventos na simulação:

- *Add*: um novo livro é adicionado pelo lado esquerdo da estante, empurrando outros livros para a direita, se necessário. Nenhum livro se move para a direita a não ser que seja empurrado por um livro adjacente (que encosta nele) à sua esquerda. Qualquer livro que não está inteiramente sobre a estante, cai pela borda direita. Nenhum livro será mais grosso que a largura da estante. Nenhum livro que está atualmente na estante será colocado novamente.
- *Remove*: se um livro está na estante, então o livro é removido, deixando um espaço vago. Se ele não está na estante, a operação deve ser ignorada.
- *End*: fim da simulação. Nesse caso imprima o conteúdo da estante.

Entrada

A entrada contém dados para uma ou mais simulações. O final da entrada é indicado por uma linha contendo -1. Cada simulação começa com um inteiro S que indica a largura da estante, $5 \leq S \leq 100$. Em seguida uma série de eventos Add e Remove. Um evento Add é uma linha que começa com ‘A’ maiúsculo, seguido pelo ID de um livro, e um inteiro W que indica a largura do livro, $0 < W \leq S$. Um evento Remove é uma linha que começa com ‘R’ maiúsculo seguido pelo ID de um livro. Finalmente, o evento final é uma linha contendo apenas um ‘E’ maiúsculo. Cada número em um evento tem um espaço em branco antes.

Saída

Para cada caso simulado, escreva uma linha contendo o número da simulação (siga exatamente o formato do exemplo de saída), seguido por uma lista dos IDs dos livros que permanecem na estante, na ordem da esquerda para a direita. Lembre-se do salto de linha no final.

Exemplo	
Entrada	Saída
10	PROBLEM 1: 15 3
R 3	PROBLEM 2:
A 6 5	PROBLEM 3: 7 6
A 42 3	
A 3 5	
A 16 2	
A 15 1	
R 16	
E	
7	
A 49 6	
A 48 2	
R 48	
E	
5	
A 1 1	
A 2 1	
A 3 1	
R 2	
A 4 1	
A 5 1	
R 5	
R 4	
A 6 1	
A 7 4	
E	
-1	

