# 1 Problema A: Gerenciamento Financeiro

Larry graduated this year and finally has a job. He's making a lot of money, but somehow never seems to have enough. Larry has decided that he needs to grab hold of his financial portfolio and solve his financing problems. The first step is to figure out what's been going on with his money. Larry has his bank account statements and wants to see how much money he has. Help Larry by writing a program to take his closing balance from each of the past twelve months and calculate his average account balance.

## 1.1 Entrada

The input will be twelve lines. Each line will contain the closing balance of his bank account for a particular month. Each number will be positive and displayed to the penny. No dollar sign will be included.

## 1.2 Saída

The output will be a single number, the average (mean) of the closing balances for the twelve months. It will be rounded to the nearest penny, preceded immediately by a dollar sign, and followed by the end-of-line. There will be no other spaces or characters in the output.

## 1.3 Exemplo de Entrada

100.00
489.12
12454.12
1234.10
823.05
109.20
5.27
1542.25
839.18
83.99
1295.01
1.75

## 1.4 Exemplo de Saída

$1581.42

## 2 Problema B: Inicie a startup

Clearly the economy is bound to pick up again soon. As a forward-thinking Internet entrepreneur, you think that the 'Net will need a new search engine to serve all the people buying new computers. Because you're frustrated with the poor results most search engines produce, your search engine will be better.

You've come up with what you believe is an innovative approach to document matching. By giving weight to the number of times a term appears in both the search string and in the document being checked, you believe you can produce a more accurate search result.

Your program will be given a search string, followed by a set of documents. You will calculate the score for each document and print it to output in the order the document appears in the input file. To calculate the score for a document you must first calculate the term score for each term appearing in the search string. A term score is the number of times a term occurs in the search string multiplied by the number of times it occurs in the document. The document score is the sum of the square roots of each term score.

### 2.1 Entrada

The input file consists of a set of documents separated by single lines containing only ten dashes, "----------". No line will be longer than 250 characters. No document will be longer than 100 lines. The first document is the search string. The input file terminates with two lines of ten dashes in a row.

The input documents will use the full ASCII character set. You must parse each document into a set of terms.

Terms are separated by whitespace in the input document. Comparisons between terms are case-insensitive. Punctuation is removed from terms prior to comparisons, e.g. "don't" becomes "dont". The resulting terms should contain only the characters {[a-z],[0- 9]}. A term in the input consisting only of punctuation should be ignored. You may assume the search string and each document will have at least one valid term.

### 2.2 Saída

The output is a series of scores, one per line, printed to two decimal places. The scores are printed in the order the documents occur in the input. No other characters may appear in the output.

### 2.3 Exemplo de Entrada

fee fi fo fum

----------

fee, fi, fo! fum!!
———-
fee fee fi, me me me
———-
———-

### 2.4 Exemplo de Saída

4.00
2.41

# 3 Problema C: Brincadeira

Alice e Beto são amigos desde crianças. Hoje em dia estão estudando na universidade, mas sempre que se encontram relembram os tempos de infância tirando par-ou-ímpar para decidir quem escolhe o filme a ser assistido, ou qual o restaurante em que vão almoçar, etc.

Ontem Alice confidenciou a Beto que ela guarda os resultados de cada vez que tiraram par- ou-ímpar desde que a brincadeira começou, no jardim de infância. Foi uma grande surpresa para Beto! Como Beto cursa Ciência da Computação, ele decidiu mostrar a Alice sua habilidade em programação, escrevendo um programa para determinar quantas vezes cada um ganhou o par-ou-ímpar no período de todos esses anos.

## 3.1 Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um único inteiro $1 \leq N \leq 10000$ que indica o número de vezes que os amigos tiraram par-ou-ímpar. A segunda linha de um caso de teste contém N inteiros Ri , separados por espaçoo, descrevendo a lista de resultados. Se Ri = 0 significa que Alice ganhou o i-ésimo jogo, se Ri = 1 significa que Beto ganhou o i-ésimo jogo ($1 \leq i \leq N$ ). O final da entrada é indicado por N = 0.

*A entrada deve ser lida da entrada padrão.*

## 3.2 Saída

Para cada caso de teste da entrada seu programa deve produzir uma linha na saída no formato 'Alice ganhou X e Beto ganhou Y', onde $X > 0$ e $Y > 0$.

*A saída deve ser escrita na saída padrão.*

## 3.3 Exemplo de Entrada

```
5
0 0 1 1 0
6
0 0 0 0 0 1
0
```

## 3.4 Exemplo de Saída

```
Alice ganhou 3 e Beto ganhou 2
Alice ganhou 5 e Beto ganhou 1
```

# 4 Problema D: Socorro!

Bem, nós temos que admitir: precisamos da ajuda de vocês. Este ano as coisas não acon- teceram como previsto, e não conseguimos terminar o software do sistema de competição em tempo. Uma parte vital está faltando, e como vocês sabem, precisamos do sistema funcionando corretamente para esta tarde. A parte do sistema que está faltando é o módulo que computa a pontuação de um time, dada a lista de submissões desse time. Socorro, socorro, alguém nos ajude por favor!

## 4.1 Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um inteiro N indicando o número de submissões do time. Cada uma das N linhas seguintes descreve uma submissão, no formato
problema minutos resultado
onde problema é uma letra de 'A' a 'Z', minutos é um inteiro representando os minutos passados desde o início da competição até o momento dessa submissão (0 lt minutos lt 300) e resultado é o resultado dessa submissão ('correto' ou 'incorreto'). As submissões estão em ordem crescente de minutos, e haverá no máximo um resultado correto para cada problema. O final da entrada é indicado por N = 0.

*A entrada deve ser lida da entrada padrão.*

## 4.2 Saída

Para cada caso de teste da entrada seu programa deve produzir uma linha na saída, contendo dois inteiros S e P , separados por espaço, onde S representa o número de problemas com o resultado correto e P é o tempo de submissão (em minutos) em que cada problema foi julgado correto, acrescido de 20 para cada submissão julgada incorreto de um problema que mais tarde foi julgado correto.

*A saída deve ser escrita na saída padrão.*

## 4.3 Exemplo de Entrada

3
A 120 incorreto
A 130 incorreto
A 200 incorreto
5
A 100 correto
B 110 incorreto
B 111 correto
C 200 correto

D 300 incorreto

### 4.4 Exemplo de Saída

0 0
3 431

# 5 Problema E: Jogo do Bicho

Em um país muito distante, as pessoas são viciadas em um jogo de apostas bastante simples. O jogo é baseado em números e é chamado jogo do bicho. O nome do jogo deriva do fato que os números são divididos em 25 grupos, dependendo do valor dos dois ultimos dígitos (dezenas e unidades), e cada grupo recebe o nome de um animal. Cada grupo é associado a um animal da seguinte forma: o primeiro grupo (burro) consiste nos números 01, 02, 03 e 04; o segundo grupo (águia) é composto dos números 05, 06, 07 e 08; e assim em diante, até o ultimo grupo contendo os números 97, 98, 99 e 00.

As regras do jogo são simples. No momento da aposta, o jogador decide o valor da aposta V e um número N ($0 \leq N \leq 1000000$). Todos os dias, na praça principal da cidade, um número M é sorteado ($0 \leq M \leq 1000000$). O prêmio de cada apostador é calculado da seguinte forma:

- se M e N têm os mesmos quatro ultimos dígitos (milhar, centena, dezena e unidade), o apostador recebe V x 3000 (por exemplo, N = 99301 e M = 19301);

- se M e N têm os mesmos três ultimos dígitos (centena, dezena e unidade), o apostador recebe V x 500 (por exemplo, N = 38944 e M = 83944);

- se M e N têm os mesmos dois ultimos dígitos (dezena e unidades), o apostador recebe V x 50 (por exemplo, N = 111 e M = 552211);

- se M e N têm os dois ultimos dígitos no mesmo grupo, correspondendo ao mesmo animal, o apostador recebe V x 16 (por exemplo, N = 82197 and M = 337600);

- se nenhum dos casos acima ocorrer, o apostador não recebe nada.

Obviamente, o prêmio dado a cada apostador é o máximo possível de acordo com as regras acima. No entanto, não é possível acumular prêmios, de forma que apenas um dos critérios acima deve ser aplicado no cálculo do prêmio. Se um número N ou M com menos de quatro dígitos for apostado ou sorteado, assuma que dígitos 0 devem ser adicionados na frente do número para que se torne de quatro dígitos; por exemplo, 17 corresponde a 0017. Dado o valor apostado, o número escolhido pelo apostador, e o número sorteado, seu programa deve calcular qual o prêmio que o apostador deve receber.

## 5.1 Entrada

A entrada contém vários casos de teste. Cada caso consiste em apenas uma linha, contendo um número real V e dois inteiros N e M , representando respectivamente o valor da aposta com duas casas decimais ($0.01 \leq V \leq 1000.00$), o número escolhido para a aposta ($0 \leq N \leq 1000000$) e o número sorteado ($0 \leq M \leq 1000000$). O final da entrada é indicado por uma linha contendo V = M = N = 0.

### 5.2 Saída

Para cada um dos casos de teste seu programa deve imprimir uma linha contendo um número real, com duas casas decimais, representando o valor do prêmio correspondente a aposta dada.

### 5.3 Exemplo de Entrada

32.20 32 213929
10.50 32 213032
2000.00 340000 0
520.00 874675 928567
10.00 1111 578311
0 0 0

### 5.4 Exemplo de Saída

515.20
5250.00
6000000.00
0.00
500.00

# 6 Problema F: Gerente de Espaço

É bem verdade que a maioria das pessoas não se importa muito com o que ocorre dentro de um computador, desde que ele execute as tarefas que devem ser desempenhadas. Existem, no entanto, alguns poucos nerds que sentem prazer em acompanhar o movimento de bits e bytes dentro da memória do computador.

É para esse público, constituído principalmente de adolescentes, que a multinacional de software ACM (Abstractions of Concrete Machines) deseja desenvolver um sistema que acom- panhe e produza um relatório das operações efetuadas em um disco rígido. Um disco rígido é composto de uma seqüência de células atômicas de armazenamento, cada uma de tamanho 1Kb.

Especificamente, a ACM deseja acompanhar três tipos de operações:

. insere NOME T insere no disco o arquivo NOME, de tamanho T. Você pode supor que um arquivo com esse nome não existe ainda no disco. O tamanho T de um arquivo é dado na forma XKb, XMb, ou XGb, onde X é um inteiro ($0 < X \leq 1023$). NOME é uma cadeia de caracteres com comprimento máximo 10.

. remove NOME remove o arquivo NOME do disco. Se um arquivo com esse nome não existe, não faz nada;

. otimiza compacta o disco, deslocando os arquivos existentes na direção do início do disco, eliminando espaços livres entre dois arquivos subseqüentes, e preservando a ordem em que os arquivos aparecem no disco, de modo a deixar um espaço de memória livre no final do disco.

A capacidade de um disco é sempre um número múltiplo de 8Kb. No início, o disco está vazio, ou seja, contém um bloco livre do tamanho da capacidade do disco. Um arquivo é sempre armazenado em um bloco de células de armazenamento contíguas. O arquivo a ser inserido deve ser sempre colocado no início do menor bloco livre cujo tamanho é maior ou igual ao tamanho do arquivo. Se mais de um bloco livre é igualmente adequado, escolha o mais próximo do começo do disco. Caso não seja possível inserir o arquivo por falta de um bloco livre suficientemente grande, deve-se executar automaticamente o comando otimiza. Se após a otimização ainda não for possível inserir o arquivo, uma mensagem de erro deve ser produzida. No caso de todas as operações serem executadas sem erro, seu programa deve produzir uma estimativa aproximada do estado final do disco, conforme descrito abaixo.

Lembre que 1Mb corresponde a 1024Kb, enquanto 1Gb corresponde a 1024Mb.

## 6.1 Entrada

A entrada é constituída de vários casos de teste. A primeira linha de um caso de teste contém um único inteiro N indicando o número de operações no disco ($0 < N \leq 10000$). A segunda linha de um caso de teste contém a descriçao do tamanho do disco, composta por um inteiro D ($0 < D \leq 1023$), seguido de um especificador de unidade; o especificador de unidade é uma cadeia de dois caracteres no formato Kb, Mb ou Gb. Cada uma das N linhas seguintes contém e a descrição de uma operação no disco (insere, remove ou otimiza, conforme descrito acima). O final da entrada é indicado por $N = 0$.

## 6.2 Saída

Para cada caso de teste seu programa deve produzir uma linha na saída. Se todas as operações de inserção forem executadas sem erro, seu programa deve produzir uma linha contendo uma estimativa aproximada do estado do disco, apresentada como se segue. Divida o número de bytes do disco em oito blocos contíguos de mesmo tamanho. Para cada um dos oito blocos seu programa deve verificar a porcentagem P de bytes livres daquele bloco, e apresentar a estimativa do estado final no formato [C][C][C][C][C][C][C][C] onde C é ' ', '-' ou '#', dependendo se $75 < P \leq 100$, $25 < P \leq 75$ ou $0 \leq P \leq 25$, respectivamente. Caso um arquivo não possa ser inserido por falta de espaço, seu programa deve produzir uma linha contendo a expressão ERRO: disco cheio; nesse caso, operações subseqüentes do caso de teste devem ser ignoradas.

## 6.3 Exemplos de Entrada

```
3
8Kb
insere arq0001 7Kb
insere arq0002 3Mb
remove arq0001
6
8Mb
insere arq0001 4Mb
insere arq0002 1Mb
insere arq0003 512Kb
remove arq0001
remove arq0002
insere arq0001 5Mb
0
```

### 6.4 Exemplos de Saída

ERRO: disco cheio
[#][#][#][#][#][#][-][ ]