594 - One Little, Two Little, Three Little Endians – 95,3%

Escrever programas que são completamente portáveis entre diferentes sistemas operacionais, diferentes versões e plataformas de hardware é uma tarefa desafiadora. Uma das principais dificuldades é resultante de decisões feitas por fabricantes de hardware sobre como eles armazenam números inteiros na memória. Pelo fato dessas representações poderem diferir de máquina para máquina, compartilhar dados binários geralmente não pode ser feito sem modificar o modo no qual o dado é armazenado ou o modo no qual esse dado é tratado por uma ou mais plataformas.

Felizmente, há um acordo (quase) universal entre os fabricantes de hardware para resolver esses problemas para inteiros com 1 byte (8-bits). Para inteiros que precisem de mais de 8-bits, como os inteiros de 2, 4 e 8 bytes disponíveis na maioria dos computadores há dois esquemas de armazenamento incompatíveis. No primeiro modelo, inteiros são armazenados como grupos de 8 bits com o byte menos significativo ocupando a posição de menor endereço na memória (considerando os endereços desse grupo de bytes) e o byte mais significativo ocupando a maior posição dessa memória. No segundo modelo ocorre justamente o contrário, inteiros são armazenados como grupos de 8 bits com o byte mais significativo ocupando a posição de menor endereço na memória e o byte mais significativo ocupando a menor posição dessa memória. A indústria da computação apelidou esses esquemas de Little Endian e Big Endian, respectivamente. Há também um acordo quase universal que inteiros com sinais são armazenados usando a representação de “complemento de dois”, e você pode assumir que esta representação será usada.

Quando um dado binário é compartilhado entre uma máquina *Little Endian* e *Big Endian*, uma conversão precisa ser executada envolvendo a inversão dos bytes de dados. Uma vez que esses dados foram invertidos então o inteiro é corretamente interpretado pelo hardware como o valor original do modelo original. O objetivo deste problema é escrever um programa que lerá uma lista de inteiros e imprimirá o inteiro equivalente na outra representação.

Entrada

A entrada consiste em uma lista de inteiros. O final do arquivo denota o final da lista de entradas. Todos os inteiros poderão ser representados como inteiros de 32 bits com sinal. Isto é, os inteiros de entrada variarão de -2147483648 a 2147483647.

Saída

Para cada inteiro de entrada imprima uma única linha no arquivo de saída. A linha deverá conter o valor de entrada seguido pela expressão “`converts to`” seguida pelo resultante da conversão.

Exemplo de Entrada

```
123456789
-123456789
1
16777216
20034556
```

Exemplo de Saída

```
123456789 converts to 365779719
-123456789 converts to -349002504
1 converts to 16777216
16777216 converts to 1
20034556 converts to -55365375
```

686 - A Conjectura de Goldbach (II) – 94%

Conjectura de Goldbach: Para cada número par n maior ou igual a 4, existe ao menos um par de números primos p_1 e p_2 tal que $n = p_1 + p_2$.

Esta conjectura não foi provada e nunca foi encontrado um contraexemplo. Ninguém está certo se a conjectura é verdadeira. Porém, até agora, para qualquer número passado foi possível encontrar um par de números primos que satisfizessem a conjectura. O problema aqui é escrever um programa que calcule a quantidade de todos os pares de primos que satisfazem a conjectura para um dado número par.

Uma sequência de números pares é dada como entrada. O programa deverá imprimir, para cada número a quantidade de pares definida acima. Note que estamos interessados no número de diferentes pares e assim, você **não** deverá contar (p_1, p_2) e (p_2, p_1) separadamente como dois pares.

Entrada

Um inteiro é dado em cada linha de entrada. Você pode assumir que cada inteiro é par e maior ou igual a 4 e menor que 2^{15}. A entrada será finalizada pelo número 0 (zero).

Saída

Cada linha de saída deverá conter apenas um número (sem espaços em branco ou outros caracteres).

Exemplo de Entrada

```
6
10
12
0
```

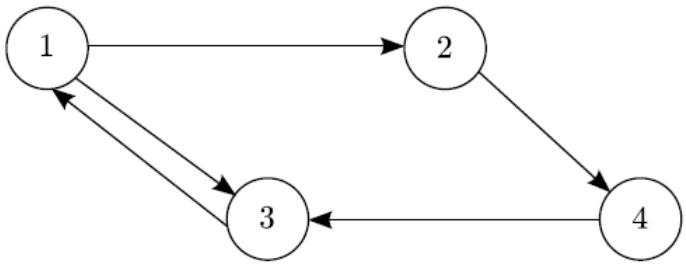
Exemplo de Saída

```
1
2
1
```

821 - Cliques em Páginas – 94,4%

Foi identificado que, na média, apenas 19 cliques são necessários para mover de uma página na web para outra. Isto é, considerando que as páginas na web são nós em um grafo, então o caminho médio entre qualquer par arbitrário de nós no grafo será 19.

Dado um grafo no qual todos os nós podem ser alcançados de um ponto inicial, seu trabalho é encontrar a **média dos caminhos** mínimos entre pares arbitrário de nós. Por exemplo, considerando o grafo ao lado. Note que os links são representados como arestas direcionadas, já que um link de uma página na web *a* para uma página *b* não implica na existência de um link de *b* para *a*.



O tamanho do menor caminho do nó 1 até os nós 2, 3 e 4 é 1, 1 e 2, respectivamente. Do nó 2 para os nós 1, 3 e 4, os caminhos mais curtos têm tamanho 3, 2 e 1. Do nó 3 para os nós 1, 2, e 4, os caminhos mais curtos têm tamanho 1, 2 e 3. Finalmente, do nó 4 para os nós 1, 2 e 3, os caminhos mais curtos têm tamanho 2, 3 e 1. A soma desses tamanhos de caminho é 1 + 1 + 2 + 3 + 2 + 1 + 1 + 2 + 3 + 2 + 3 + 1 = 22. Já que há 12 possíveis pares de nós a serem considerados, nós obtemos o menor caminho médio como sendo 22/12, ou 1.833 (com precisão de três casas decimais).

Entrada

A entrada conterá múltiplos casos de teste. Cada caso de teste consiste em um número arbitrário de pares de inteiros, a e b, cada um representando um link da página numerada *a* para outra página numerada *b*. O número das páginas sempre variará de 1 até 100. A entrada para cada caso de teste terminará com um par de zeros, que não deverá ser tratado como número de páginas. Um par de zeros adicional seguirá o último caso de testes, representando efetivamente um caso de testes sem links, que também não deverá ser processado. O grafo não conterá links de autorreferência (isto é, links de uma página para ela mesma), e haverá ao menos um caminho de cada nó no grafo para qualquer outro nó no grafo.

Saída

Para cada caso de teste, determine o tamanho da média dos caminhos mínimos entre todos os pares de nós, com precisão de três casas decimais. Imprima o tamanho e o identificador do caso de teste (eles deverão ser numerados sequencialmente a partir de 1) de maneira similar à mostrada no exemplo de saída abaixo.

Exemplo de Entrada

```
1 2 2 4 1 3 3 1 4 3 0 0
1 2 1 4 4 2 2 7 7 1 0 0
0 0
```

Exemplo de Saída

```
Case 1: average length between pages = 1.833 clicks
Case 2: average length between pages = 1.750 clicks
```

834 – Frações Continuadas – 95,2%

Sendo b_0, b_1, b_2, ..., b_n inteiros com $b_k > 0$ para $k > 0$. A fração continuada de ordem n com coeficientes b_1, b_2, ..., b_n e termo inicial b_0 é definida pela seguinte expressão:

$$b_0 + \cfrac{1}{b_1 + \cfrac{1}{b_{2+\ldots+\frac{1}{b_n}}}}$$

que pode ser abreviado como: $[b_0; b_1, \ldots, b_n]$.

Um exemplo de fração continuada de ordem n=3 é: [2;3,1,4]. Isto é equivalente a:

$$2 + \cfrac{1}{3 + \cfrac{1}{1+\frac{1}{4}}} = \frac{43}{19}$$

Escreva um programa que determine a expansão de um dado número racional como uma fração continuada. Para garantir uma única representação de cada número b_n deverá ser maior que 1.

Entrada

A entrada consistirá em um número não determinado de números racionais. Cada número racional será definido por dois inteiros, numerador e denominador.

Saída

Para cada número racional dado na entrada, você deverá imprimir o número correspondente em fração continuada.

Exemplo de Entrada

```
43 19
1 2
```

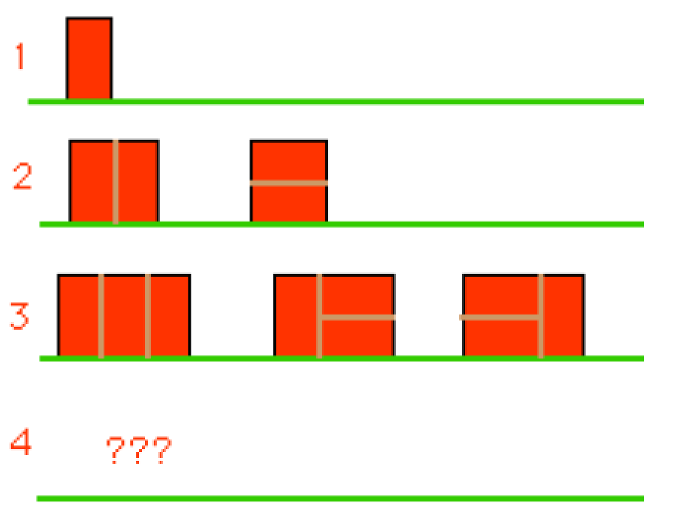
Exemplo de Saída

```
[2;3,1,4]
[0;2]
```

900 – Padrões no Muro de Tijolos – 94,9%

Imagine que nós queremos construir um muro de tijolos normais, cuja altura é duas vezes sua largura. Se o muro tiver duas unidades de altura, nós podemos fazer nosso muro seguindo um diferente número de padrões, dependendo do comprimento que quisermos neste muro. Da figura abaixo, podemos observar que:

- Há apenas um padrão possível se o muro tiver uma unidade de comprimento: um tijolo em pé;
- Há dois padrões para um muro de comprimento 2: dois tijolos em pé ou dois tijolos deitados (um sobre o outro);
- Há três padrões possíveis para um muro de comprimento 3.
- Quantos padrões há para um muro de tamanho 4? E, para um muro de tamanho 5?



Problema

Você ficou encarregado de escrever um programa que, dado o comprimento do muro, determine quantos padrões diferentes existem para um muro com esse comprimento.

Entrada

Seu programa receberá uma sequência de inteiros positivos, um por linha, cada um representando o comprimento de um muro. O valor máximo para o comprimento do muro é 50. A entrada termina com um valor 0 (zero).

Saída

Para cada comprimento de muro da entrada, seu programa deverá imprimir como saída o número de diferentes padrões correspondentes à entrada. Imprima um valor por linha.

Exemplo de Entrada

```
1
2
3
20
40
0
```

Exemplo de Saída

```
1
2
3
10946
165580141
```

924 - Espalhando as Novidades – 94%

Em uma grande organização, todos sabem muito sobre seus colegas. No entanto, relações de amizade são mantidas apenas com alguns deles, para os quais as novidades são contadas.

Suponha que sempre que um empregado sabe uma novidade, ele a conta para todos os seus amigos no dia seguinte. Então, no primeiro dia, o empregado 'fonte da informação' contará a novidade a todos os seus amigos; no segundo dia, os amigos contarão a seus amigos e assim por diante.

O seu objetivo é determinar:

- **o tamanho do boom máximo da notícia**: que é o maior número de empregados que ouviu a novidade pela primeira vez em um único dia; e
- **o primeiro dia onde ocorreu o boom máximo**: que é o primeiro dia no qual o boom máximo ocorreu.

O Problema

Escreva um programa que, dadas as relações de amizade entre empregados e o empregado 'fonte da informação', compute os valores do tamanho do boom máximo e o primeiro dia no qual esse boom máximo ocorreu.

Entrada

A primeira linha da entrada conterá o número de empregados E ($1 \leq E \leq 2500$). Os empregados serão numerados de 0 até E-1.

Cada uma das E linhas seguintes especificará o conjunto de amigos de um empregado (do empregado 0 até o empregado E-1). O conjunto de amigos conterá o número de amigos N ($0 \leq N \leq 15$), seguido por N inteiros distintos representando os amigos do empregado. Todos estes inteiros serão separados por um espaço em branco.

A próxima linha conterá um inteiro T ($1 \leq T < 60$), que é o número de casos de teste.

Cada uma das T linhas seguintes conterá um empregado, que representará o (único) 'fonte de informação' para aquele caso de teste.

Saída

A saída consistirá em T linhas, uma para cada caso de teste.

Se nenhum empregado (além da fonte) ouvir a novidade, a linha de saída deverá conter apenas o inteiro 0 (zero).

Caso contrário, a saída conterá dois inteiros M e D, separados por um espaço em branco, onde M é o valor do tamanho do boom máximo e D é o primeiro dia no qual esse boom máximo ocorreu.

Exemplo de Entrada

```
6
2 1 2
2 3 4
3 0 4 5
1 4
0
2 0 2
3
0
4
5
```

Exemplo de Saída

```
3 2
0
2 1
```

1586 - Massa Molar – 93,4%

Um composto orgânico é qualquer membro de uma grande classe de compostos cujas moléculas contêm carbono. A massa molar de compostos orgânicos é a massa de uma molécula desse composto. Essa massa molar pode ser computada a partir dos pesos atômicos padrões de seus elementos.

Dado um composto orgânico definido por sua fórmula, o doutor Chon quer descobrir sua massa molar. Uma fórmula molecular, como $C_3H_4O_3$, identifica cada elemento constituinte da molécula utilizando o símbolo químico dos elementos que a compõe. Se uma molécula contém mais de um átomo de algum elemento, esta quantidade é indicada usando um número em subscrito após o símbolo químico.

Para este problema, assuma que a fórmula molecular é representada por apenas quatro elementos: C (carbono), H (hidrogênio), O (oxigênio), e N (nitrogênio). A seguinte tabela apresenta o peso atômico padrão destes elementos:

Atomic Name	Carbon	Hydrogen	Oxygen	Nitrogen
Standard Atomic Weight	12.01 g/mol	1.008 g/mol	16.00 g/mol	14.01 g/mol

Dada uma fórmula molecular, escreva um programa que compute a massa molar da respectiva fórmula.

Entrada

A entrada consiste em T casos de teste. O número de casos de teste T é dado na primeira linha da entrada. Cada caso de teste contém uma única linha, a qual possui a fórmula molecular como uma string. Cada símbolo químico é dado por uma letra maiúscula e o tamanho da string será maior que 0 (zero) e menor que 80. A quantidade de átomos n associada a cada elemento químico será omitida quando houver apenas um átomo do respectivo elemento, para os demais casos esta quantidade será representada por um número após o símbolo do elemento e este número valerá entre 2 e 99.

Saída

Seu programa deverá imprimir uma linha por caso de teste. A linha deverá conter a massa molar da respectiva fórmula molecular. **Observe as casas decimais utilizadas no exemplo de saída**.

Exemplo de Entrada

```
4
C
C6H5OH
NH2CH2COOH
C12H22O11
```

Exemplo de Saída

```
12.010
94.108
75.070
342.296
```