156 – Anagramas – 94%

Muitos fãs de quebra-cabeças com palavras estão acostumados com anagramas – grupos de palavras formadas pelas mesmas letras em ordens diferentes – por exemplo OPTS, SPOT, STOP, POTS e POST. Algumas palavras, no entanto, não possuem este atributo, não importa como você rearranje suas letras, elas não formarão outra palavra. Estas palavras são chamadas ananagramas, um exemplo é QUIZ.

Obviamente, estas definições dependem do domínio no qual você está trabalhando; você pode achar que ATHENE é um ananagrama, porém qualquer químico vai facilmente rearranjar suas letras e formar ETHANE. Um domínio possível seria todas as palavras da língua inglesa, mas isto pode levar a alguns problemas. Por outro lado, alguém poderia restringir o domínio, digamos, para música, no qual a palavra SCALE se tornará um "ananagrama relativo" (LACES não está no mesmo domínio), mas a palavra NOTE não é um ananagrama pois podemos produzir TONE.

Escreva um programa que lerá um dicionário de um domínio restrito e determine os ananagramas relativos. Note que palavras de uma única letra são ananagramas relativos, já que não é possível formar uma palavra diferente com apenas esta letra. O dicionário não conterá mais de 1000 palavras.

Entrada

A entrada consistirá em uma série de linhas. Nenhuma linha terá mais de 80 caracteres, mas poderá conter qualquer número de palavras. Palavras contêm uma ou mais letras (limitadas a 20 letras) maiúsculas e/ou minúsculas e nenhuma palavra estará quebrada entre duas linhas. Haverá um ou mais espaços ao redor das palavras. Note que duas palavras que contém as mesmas letras, mas diferindo apenas se as letras estão em maiúscula e minúscula são consideradas anagramas uma da outra, desta forma tIeD e EdiT são anagramas. O arquivo terminará com uma linha contendo apenas a palavra #.

Saída

A saída consistirá em uma séria de linhas. Cada linha conterá uma única palavra que é um ananagrama relativo em relação as demais palavras do dicionário. As palavras devem ser impressas em ordem lexicográfica (ordem alfabética, mas primeiro as maiúsculas e depois as minúsculas). Sempre haverá ao menos um ananagrama relativo.

Exemplo de Entrada

```
ladder came tape soon leader acme RIDE lone Dreis peat
ScAlE orb eye Rides dealer NotE derail LaCeS drIed
noel dire Disk mace Rob dries
#
```

Exemplo de Saída

```
Disk
NotE
derail
drIed
eye
ladder
soon
```

299 - Troca de Vagões – 94,4%

Em uma velha ferrovia, talvez você ainda encontre um dos últimos “rearranjadores de trens”. Um rearranjador de trem é um empregado da ferrovia responsável unicamente por rearranjar os vagões de um trem.

Uma vez que os vagões estejam organizados na melhor ordem possível, tudo que o maquinista precisará fazer é deixar os vagões, um por um, em seus respectivos destinos.

O título de “rearranjador de trem” vem da primeira pessoa que executou esta tarefa, em uma estação próxima a uma ponte. Ao invés de abrir verticalmente, essa ponte rodava ao redor de seu pilar posicionado no centro do rio. Após rotacionar 90 graus, barcos podiam passar a esquerda e a direita.

O primeiro rearranjador de trem descobriu que a ponte poderia operar com até dois vagões em cima dela. Ao rotacionar a ponte 180 graus, os vagões trocavam de ordem, permitindo o rearranjo dos vagões (como um efeito colateral, após o rearranjo os vagões estariam apontando para o sentido oposto, mas já que os vagões podem se mover para ambos os sentidos, isso não era um problema).

Agora que quase todos os arranjadores de trens já morreram, a companhia ferroviária gostaria de automatizar esse tipo de operação. Você deverá desenvolver um programa que decidirá, para um dado trem, o número mínimo de trocas de vagões adjacentes que serão necessárias para ordenar o trem.

Especificação da Entrada

A entrada conterá na primeira linha o número de testes a serem feitos (N). Cada teste consiste em duas linhas de entrada. A primeira linha do teste contém um inteiro L que determina o tamanho do trem ($0 \leq L \leq 50$). A segunda linha conterá uma permutação dos números de 1 até L, indicando a ordem atual dos vagões. Os vagões devem ser ordenados em ordem crescente.

Especificação da Saída

Para cada teste você deverá imprimir a sentença: '`Optimal train swapping takes` *S* `swaps.`' onde *S* é um inteiro correspondente ao menor número de trocas.

Exemplo de Entrada

```
3
3
1 3 2
4
4 3 2 1
2
2 1
```

Exemplo de Saída

```
Optimal train swapping takes 1 swaps.
Optimal train swapping takes 6 swaps.
Optimal train swapping takes 1 swaps.
```

440 – Uni, Duni, Tê – 94,5%

Certamente você já teve a experiência de usar a internet quando muitas pessoas estavam usando ao mesmo tempo e ela se tornou muito, muito lenta.

Para acabar com esse problema, a Universidade de Ulm desenvolveu um esquema de contingência para os horários de pico, para cortar o acesso à rede de algumas cidades do país de maneira sistemática e totalmente justa. As cidades alemãs foram enumeradas aleatoriamente de 1 até n. Freiburg era número 1, Ulm era número 2, Karlsruhe era número 3 e assim por diante.

Então, o número m deveria ser sorteado aleatoriamente e o acesso à internet deveria ser cortado primeiro na cidade 1, e então em cada m-ésima cidade depois disto. A contagem deve continuar no início da lista após atingir a cidade n, além de ignorar as cidades já desligadas.

Por exemplo, se n=17 e m=5, o corte ao acesso à internet deve ser feito na seguinte ordem: [1,6,11,16,5,12,2,9,17,10,4,15,14,3,8,13,7]. O problema é que a cidade Ulm deveria ser sempre a última a ter a internet desligada (afinal de contas, é de lá que todos os programadores vem), então, para um dado n, o número "aleatório" m precisa ser cuidadosamente escolhido de forma que a cidade 2 seja a última a ser selecionada.

Seu trabalho é escrever um programa que lerá o número de cidades (n) e então determinará o menor inteiro m de forma a garantir que Ulm (cidade número 2) será a última a ter a internet cortada.

Especificação de Entrada

Cada entrada conterá uma ou mais linhas, cada linha conterá um inteiro n com 3 <= n < 150, representando o número de cidades do país. A entrada será encerrada com o valor 0 (zero) para n.

Especificação de Saída

Para cada linha de entrada, imprima uma linha contendo o inteiro m que satisfaça os requisitos especificados acima.

Exemplo de Entrada

```
3
4
5
6
7
8
9
10
11
12
0
```

Exemplo de Saída

```
2
5
2
4
3
11
2
3
8
16
```

446 - Kibbles ‘n’ Bits ‘n’ Bits ‘n’ Bits – 93,3%

Um certo programador exausto está escrevendo um programa que recebe dois números de uma vez na forma hexadecimal, executa uma adição ou subtração com eles, e imprime o resultado na forma decimal. No entanto, a representação binária dos números hexadecimais também precisa ser impressa, da exata maneira apresentada pelo exemplo abaixo.

Este programador ficaria muito satisfeito em desenvolver essa rotina, porém, toda vez que ele tenta implementar algo na base 2, ele tem urticárias. Então, se você desenvolver este pequeno programa para ele, ele ficará eternamente grato.

Você poderá assumir o seguinte:

- O maior número hexadecimal será `FFF`.
- Nas subtrações, o segundo número sempre será menor do que o primeiro (não haverá resultados negativos).
- O espaçamento na entrada será totalmente uniforme, isto é, não haverá espaços em branco no início ou final de uma linha, e apenas um espaço em branco entre cada elemento.

Entrada

A entrada do seu programa virá no seguinte formato:
N (este é o número de expressões a serem computadas)
*HEX*1 (+ ou -) *HEX*2 (primeira expressão)
.
.
.
*HEX*1 (+ ou -) *HEX*2 (enésima expressão)
Onde HEX1 é o primeiro número no formado hexadecimal, seguido por um espaço em branco, o sinal + ou -, seguido por um espaço em branco e o segundo número em hexadecimal.

Saída

A saída deverá ser apresentada no seguinte formado, contendo os N resultados referentes a entrada:
*BINARY*1 (+ ou -) *BINARY*2 = *DECIMAL* (primeiro resultado)
.
.
.
*BINARY*1 (+ ou -) *BINARY*2 = *DECIMAL* (enésimo resultado)

Exemplo de Entrada

```
2
A + 3
AAA + BBB
```

Exemplo de Saída

```
0000000001010 + 0000000000011 = 13
0101010101010 + 0101110111011 = 5733
```

477 – Pontos em Figuras: Retângulos e Círculos – 94,7%

Dada uma lista de figuras (retângulos e círculos) e uma lista de pontos no plano x-y, determine, para cada ponto, quais figuras contêm o ponto (pode ser que nenhuma figura contenha).

Entrada

Haverá n <= 10 descrições de figuras, uma por linha. O primeiro caractere determinará o tipo da figura ("r" para retângulo e "c" para círculo). Este caractere será seguido pelos valores descritos abaixo:

- Para um retângulo, haverá quatro valores reais designando as coordenadas x-y dos cantos superior esquerdo e inferior direito da figura.
- Para um círculo, haverá três valores reais, designando a coordenada x-y do centro do círculo e o raio do círculo.

O final da lista de figuras será determinado por um asterisco (*) no início da linha.

As demais linhas conterão as coordenadas x-y, uma por linha, dos pontos a serem testados. O final desta lista será indicado por um ponto com coordenada 9999.9 9999.9; estes últimos valores não deverão ser incluídos na saída. **Pontos que coincidam com a borda da figura não deverão ser considerados pontos internos.**

Saída

Para cada ponto testado, escreva uma mensagem na forma: ***Point i is contained in figure j*** para cada figura que contenha o ponto (pode haver mais de uma figura). Se o ponto não está contido em nenhuma figura, escreva uma mensagem na forma: ***Point i is not contained in any figure*** - Pontos e figuras devem ser numerados na ordem que eles aparecem na entrada (começando por 1, ver exemplos de entrada e saída).

Exemplo de Entrada

```
r 8.5 17.0 25.5 -8.5
c 20.2 7.3 5.8
r 0.0 10.3 5.5 0.0
c -5.0 -5.0 3.7
r 2.5 12.5 12.5 2.5
c 5.0 15.0 7.2
*
2.0 2.0
4.7 5.3
6.9 11.2
20.0 20.0
17.6 3.2
-5.2 -7.8
9999.9 9999.9
```

Exemplo de Saída

```
Point 1 is contained in figure 3
Point 2 is contained in figure 3
Point 2 is contained in figure 5
Point 3 is contained in figure 5
Point 3 is contained in figure 6
Point 4 is not contained in any figure
Point 5 is contained in figure 1
Point 5 is contained in figure 2
Point 6 is contained in figure 4
```

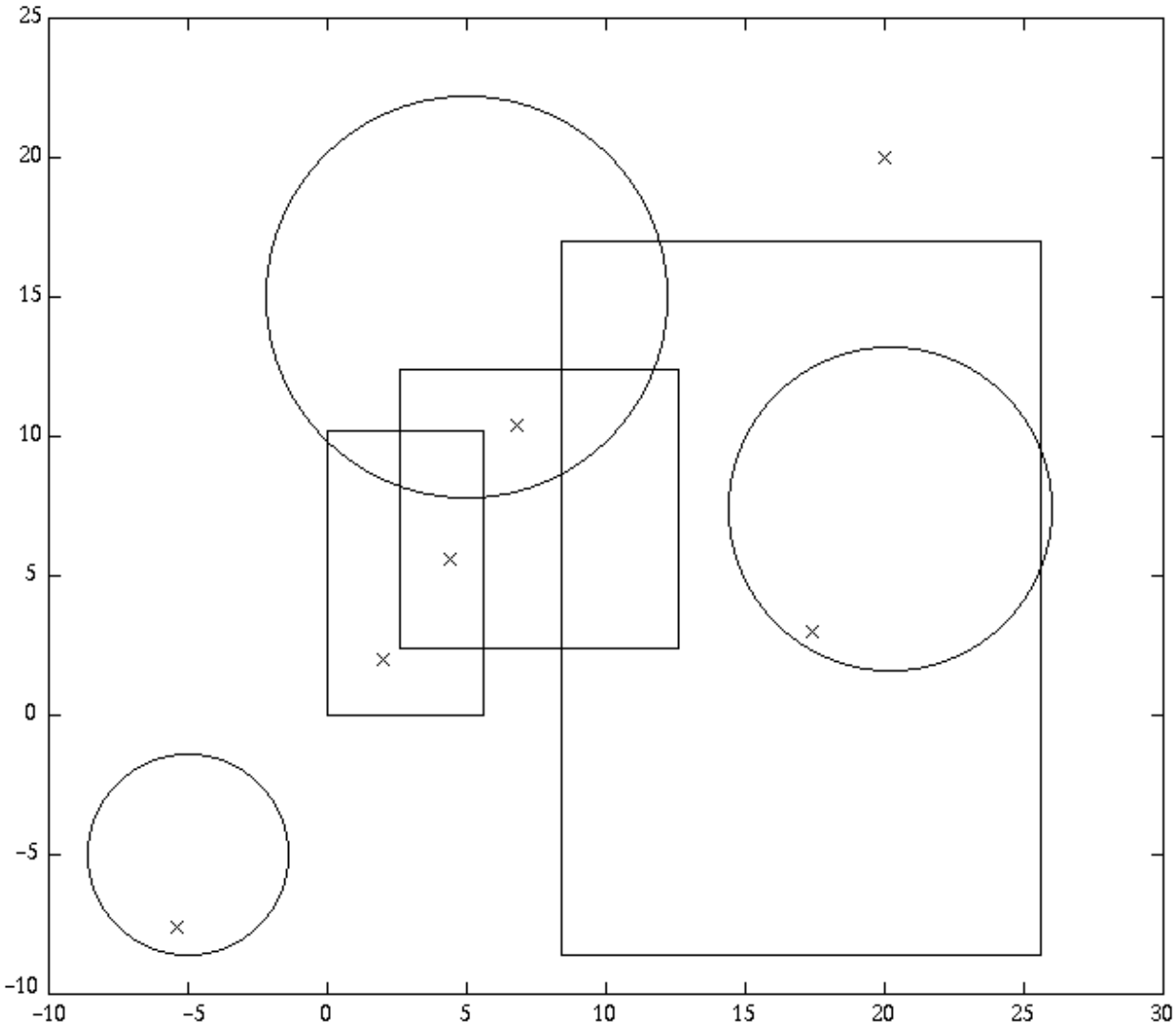

Diagrama com o exemplo de figuras e pontos

499 – Qual é a Frequência? – 94,3%

```
#include <stdio.h>
main()
{
int i;
char *suffix[]= { "st", "nd", "rd" };
char *item[]= { "Unix" , "cat", "sed", "awk", "grep", "ed", "vi"};

printf("In the beginning, there was nothing.\n");
for (i= 0; i < 7; i++)
printf("And on the %d%s day, God created %s. And it was good.\n",
i + 1, (i < 3) ? suffix[i] : "th", item[i]);
}
```

E então Deus viu que o programa `vi` conduziu as pessoas à tentação. Ao invés de escolherem as maneiras mais corretas do `make`, `dbx`, e `RCS`, as pessoas usaram longas linhas de comando `printf()` e backups em fitas.

Então Deus decretou “Eu vejo aqueles Engenheiros que corromperam meu `vi`.” e também “Eu criarei *emacs*, um editor mais poderoso do que palavras”. Além disso, para cada instanciação do `vi` até o momento, o Engenheiro responsável irá fazer uma penitência. E mais, a penitência será dolorosa; haverá muita lamúria e ranger de dentes. O Engenheiro lerá muitas linhas de texto. Para cada linha de texto, ele deverá dizer quais letras ocorrem com maior frequência.”

“E eu deixo a todos com a Minha Regra de Ouro: 'Amigos não deixarão amigos usarem vi'."

Entrada e Saída

Cada linha da saída deverá conter uma lista das letras que ocorreram com a frequência máxima na linha de entrada correspondente, seguidas pela frequência.

A lista de letras deve ser ordenada alfabeticamente, porém primeiro todas as letras maiúsculas e depois todas as minúsculas. Caso apenas uma letra possua frequência máxima, somente ela e sua frequência deverão ser impressas.

Exemplo de Entrada

```
When riding your bicycle backwards down a one-way street, if the
wheel falls of a canoe, how many ball bearings does it take to fill
up a water buffalo?
Hello Howard.
```

Exemplo de Saída

```
e 6
al 7
a 3
Hlo 2
```

1124 – Jeopardy de Celebridades – 94,6%

É difícil construir um problema que seja simples o suficiente para todos entenderem e ainda assim difícil o bastante para merecer algum respeito. Tipicamente nós falhamos de um lado ou do outro. O quão simples um problema poderá realmente ser?

Aqui, assim como no Jeopardy de Celebridades, questões e respostas são um pouco confusas, e, pelo fato dos participantes serem celebridades, há uma necessidade real de fazer o desafio simples. Seu programa deverá preparar uma questão a ser resolvida – uma equação a ser resolvida - a partir de uma resposta. Especificamente, você deverá escrever um programa que encontrará a equação mais simples possível que quando resolvida encontrará a resposta dada, considerando todas as equações possíveis usando as operações matemáticas padrões. Neste contexto, "a mais simples" pode ser definida de maneira não ambígua de diferentes formas levando todos à mesma resolução. Para este problema, encontre a equação cuja transformação na resposta desejada requer menos esforço.

Por exemplo, dada a resposta X=2, você pode criar a equação 9 – X = 7. Alternativamente, você poderá construir um sistema tal que X > 0 e X^2 = 4. Estas podem não ser as equações mais simples possíveis. Resolver estes "desafios mentais" pode ser difícil para celebridades.

Entrada

Cada linha de entrada conterá uma solução na forma < símbolo > = < valor >

Saída

Para cada linha de entrada, imprima o sistema mais simples de equações que irão prover a solução, respeitando o espaçamento exatamente como dado na entrada.

Exemplo de Entrada

```
Y = 3
X=9
```

Exemplo de Saída

```
Y = 3
X=9
```