

Lisp é melhor que Java, C e CPP

Acredite ou não, esse foi o resultado de um estudo conduzido por Ron Garret (Erann Gat) no início do século. A motivação de Garret foi um outro estudo, feito por Lutz Prechelt e publicado na Communications of the ACM, que comparava a performance de tempo de execução e uso de memória de programas escritos em C, C++ e Java. Porém, diferentemente de benchmarks tradicionais, Prechelt comparou diferentes implementações de uma mesma tarefa feita por 38 desenvolvedores diferentes (em experiência e conhecimento). O estudo de Prechelt mostrou que Java é de 3 a 4 vezes mais lento que C ou C++, porém a variação maior ocorreu entre os programadores, não entre as linguagens, sugerindo que é melhor gastar mais tempo treinando os desenvolvedores do que discutindo que linguagem deve ser escolhida.

Anos depois Garret estendeu esse estudo adicionando Lisp como uma das implementações possíveis para o problema, e dessa vez, além de considerar todos os fatores de comparação de Prechelt, acrescentou o tempo de desenvolvimento como métrica. Os resultados de Garret foram surpreendentes: Lisp ganhou disparado em todos os quesitos, necessitando de menos tempo e linhas de código, consumindo menos memória e executando mais rápido que os programas feitos em C, C++ ou Java. Essa é a sua chance de provar que o estudo de Garret está errado. Como? Resolvendo o mesmo problema proposto, em menos tempo e com implementações mais rápidas. O problema que foi a base de ambos os estudos é o seguinte: Considere o seguinte mapeamento entre letras e dígitos:

```
E | J N Q | R W X | D S Y | F T | A M | C I V | B K U | L O P | G H Z
e | j n q | r w x | d s y | f t | a m | c i v | b k u | l o p | g h z
0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Queremos usar esse mapeamento para codificar números de telefone em palavras de forma que seja fácil decorá-los. Sua tarefa é escrever um programa que ache, dado um número de telefone, todas as possíveis codificações do mesmo em palavras. Um número de telefone é uma string arbitrária contendo apenas hífen (-), barras (/), e dígitos. As barras e hífen não devem ser codificados. As palavras são tiradas de um dicionário informado em ordem alfabética. Você deve imprimir apenas as palavras que codifiquem completamente o número de telefone. As palavras no dicionário podem ter letras maiúsculas e minúsculas, hífen (-) e aspas ("), porém você deve usar apenas as letras para codificar um número. A palavra deve ser impressa como foi dada no dicionário. A codificação de um número de telefone pode consistir de uma ou mais palavras, separadas por espaço. A codificação é construída palavra por palavra, da esquerda para a direita. Se, em um dado ponto da codificação nenhuma palavra do dicionário pode ser inserida, então um único dígito do número de telefone pode ser usado para a codificação, porém dois números consecutivos não são permitidos numa codificação válida. Em outras palavras: em uma codificação parcial que cobre k dígitos, o dígito $k + 1$ é codificado por ele mesmo se e somente se, primeiro, o dígito k não foi codificado por um dígito e, segundo, não existe palavra no dicionário que pode ser usada na codificação começando no dígito $k + 1$.

Entrada

Cada instância é composta por uma linha contendo um número inteiro $0 < n \leq 75000$, o número de palavras no dicionário. As próximas n linhas contêm palavras com no máximo 50 caracteres. Depois do dicionário segue um inteiro $1 < t < 100000$, e nas t linhas seguintes os números de telefone a serem codificados. Quando n for 0 seu programa deve parar.

Saída

Para cada instância seu programa deve imprimir uma linha contendo Instancia k, onde k é o numero da k-ésima instância. Para cada número de telefone processado seu programa deve imprimir todas as codificações possíveis em ordem lexicográfica crescente. Cada codificação deve ser impressa no seguinte formato: o número do telefone seguido de dois pontos (:), um espaço e a codificação.

Exemplo

Entrada:

23
Bo"
Boot
Fee
Fest
Mix
Mixer
Name
Ort
Tor
Torf
Wasser
an
blau
bo"s
da
fern
fort
je
jemand
mir
neu
o"d
so
8
112
5624-82
4824
0721/608-4067
10/783--5
1078-913-5
381482
04824
0

Saída:

Instancia 1
5624-82: Mix Tor
5624-82: mir Tor
4824: Tor 4
4824: Torf
4824: fort
10/783--5: je Bo" da
10/783--5: je bo"s 5
10/783--5: neu o"d 5
381482: so 1 Tor

04824: 0 Tor 4

04824: 0 Torf

04824: 0 fort