Maratona de Programação
1999
Caderno de Problemas
Categoria - Sênior

Problema 1: Matrizes Incas

```
Arquivo:  inca.c ou inca.pas
Entrada:  inca.in
Saída:  inca.out
```

Os incas ficaram conhecidos pela grande civilização que reinou na região dos Andes durante vários séculos. O que pouca gente sabe é que os incas construíram pirâmides de base quadrada em que a única forma de se atingir o topo era seguir em espiral pela borda, que acabava formando uma escada em espiral. Estas pirâmides ainda se encontram escondidas na floresta amazônica e sua descoberta trará uma aplicação para este exercício.

Neste problema você deverá fazer um programa que recebe uma matriz quadrada $A_{n\times n}$ de números inteiros e verifica se a matriz é inca, ou seja, se partindo do canto superior esquerdo da matriz, no sentido horário, em espiral, a posição seguinte na ordem é o inteiro consecutivo da posição anterior.

Exemplo: A matriz abaixo é inca:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 12 & 13 & 14 & 5 \\ 11 & 16 & 15 & 6 \\ 10 & 9 & 8 & 7 \end{pmatrix}$$

Entrada

São dadas várias instâncias. A primeira linha de cada instância contém a dimensão $0 \le n \le 100$ da matriz. A seguir, em cada uma das próximas n linhas são dados n inteiros correspondentes às entradas da matriz naquela linha. A última linha do arquivo de entrada contém um 0.

Saída

Você deverá imprimir um cabeçalho indicando o número da instância que está tratando (`Instancia #i`) e na linha seguinte a mensagem de que a matriz é ou não inca. As instâncias deverão estar separadas por uma linha em branco.

Exemplo

Entrada

```
4
1 2 3 4
12 13 14 5
11 16 15 6
10 9 8 7
3
1 2 3
4 5 6
7 9 8
0
```

Saída

```
Instancia #1
A matriz e inca.

Instancia #2
A matriz nao e inca.
```

Problema 2: Palavras mais comuns

```
Arquivo:  palavras.c ou palavras.pas
Entrada:  palavras.in
Saída:  palavras.out
```

O objetivo do programa que você fará é resolver o seguinte problema:

> Dado um arquivo texto e um inteiro positivo k, imprimir as k palavras que mais ocorrem no texto (e o número de vezes que elas ocorrem) em ordem decrescente de freqüência.

O problema ainda está um tanto quanto vago. Vamos torná-lo mais preciso. Suponhamos que uma *palavra* é uma seqüência de uma ou mais letras consecutivas; “Maratona” é uma palavra enquanto que “pau-brasil” não é. A seqüência de letras deve ser maximal, significando que a seqüência não pode ser aumentada sem que incluamos uma não-letra. Letras maiúsculas serão consideradas equivalentes a letras minúsculas, logo as palavras “Maratona”, “MARATONA” e “maratona” são idênticas e deverão ser consideradas como sendo a mesma palavra.
O problema ainda não está bem definido pois um arquivo pode ter mais que k palavras, todas ocorrendo com a mesma freqüência; ou ainda pode nem conter k palavras distintas. Sejamos mais precisos: as palavras mais freqüentes deverão ser impressas em ordem decrescente de freqüência, com palavras de mesma freqüência listadas em ordem alfabética. A impressão deverá parar depois que k palavras foram impressas; se mais do que k palavras estão presentes no texto. Você pode supor que o comprimento máximo de uma palavra é 31 caracteres e que no texto não ocorrem mais do que 1024 palavras distintas. Você ainda pode supor que as palavras do texto não estão acentuadas. Assim no texto escreveremos, por exemplo, “`frequente`” ao invés de “`freqüente`”, “`estao`” ao invés de “`estão`” e “`inves`” ao invés de “`invés`”.

Entrada

A primeira linha do arquivo de entrada contém um número inteiro positivo (possivelmente precedido de espaços em branco). Este número será o valor de k. A partir da segunda linha começa o texto que termina com o final do arquivo.

Saída

O arquivo de saída deve conter k linhas — uma linha para cada uma das palavras mais freqüentes no texto — se k for menor ou igual que o número t de palavras no texto. Caso contrário a saída deve conter t linhas. A i-ésima linha do arquivo de saída deve conter o número de ocorrências da i-ésima palavra mais freqüente no texto, seguido de um espaço em branco e da palavra.

Exemplo 1

Entrada

```
7
O problema ainda esta um tanto quanto vago, vamos torna-lo mais preciso.
Suponhamos que uma palavra e uma sequencia de uma ou mais letras
consecutivas; ``Maratona'' e uma palavra enquanto que ``pau-brasil'' nao e.
```

Saída

```
4 uma
3 e
2 mais
2 palavra
2 que
1 ainda
1 brasil
```

Exemplo 2

Entrada

```
20
O objetivo do programa que voce fara e resolver o seguinte problema:

Dado um arquivo texto e um inteiro positivo k, imprimir as k palavras que mais
ocorrem no texto (e o numero de vezes que elas ocorrem) em ordem decrescente
de frequencia.

O problema ainda esta um tanto quanto vago. Vamos torna-lo mais preciso.
Suponhamos que uma palavra e uma sequencia de uma ou mais letras
consecutivas; ``Maratona'' e uma palavra enquanto que ``pau-brasil'' nao e. A
sequencia de letras deve ser maximal, significando que a sequencia nao pode
ser aumentada  sem que incluamos uma nao-letra. Letras maiusculas serao
consideradas equivalentes a letras minusculas, logo as palavras ``Maratona'',
``MARATONA'' e ``maratona'' sao identicas e deverao ser consideradas como
sendo a mesma palavra.
```

Saída

```
8 e
7 que
5 uma
4 a
4 de
4 letras
4 maratona
4 o
3 mais
3 nao
3 palavra
3 sequencia
3 ser
3 um
2 as
2 consideradas
2 k
2 ocorrem
2 palavras
2 problema
```

Problema 3: Entrega de pizzas

`Arquivo: mchut.c ou mchut.pas`
`Entrada: mchut.in`
`Saída: mchut.out`

O McHut é uma nova cadeia de pizzas com um diferente sistema de entrega em domicílio. Ao contrário das pizzarias convencionais, os entregadores da McHut não ficam nas pizzarias, mas nas ruas, esperando para fazer as entregas. Ao receber uma chamada, o gerente decide que entregador deve fazer a entrega, levando em conta que o entregador deverá se deslocar inicialmente a uma filial da McHut e posteriormente até o local da chamada. O entregador escolhido deverá ser aquele para o qual a distância total percorrida é mínima. Após atender uma chamada o entregador permanece no local da entrega da pizza até ser alocado a outra entrega.
Ainda mais diferenciado é o fato de que nem todas as filiais da McHut têm o mesmo cardápio. Cada filial faz um conjunto diferente de pizzas, tornando o trabalho do gerente ainda mais complicado. Outra restrição da McHut é que em cada chamada só é permitido pedir uma pizza.
Sua tarefa neste exercício será fazer um programa de auxílio ao gerente da McHut na alocação dos entregadores às chamadas a serem realizadas.

Entrada

Na primeira linha é dado o número n $(0 < n \leq 20)$ de entregadores. Nas n linhas seguintes, são dadas as posições (ou seja, pares de inteiros indicando a posição no plano ocupada) dos entregadores. A seguir é dado o número $0 < m \leq 10$ de filiais da McHut. Em cada uma das m linhas seguintes é dada a posição de uma filial, o número de sabores que compõe seu cardápio, e uma lista de strings com os sabores feitos naquela filial. Um sabor de pizza nunca excede 20 caracteres. Em cada uma das k linhas seguintes é dada a posição de uma entrega e o sabor da pizza pedida.

Saída

A saída deverá conter k linhas (uma linha para cada chamada). Cada linha deve conter uma mensagem identificando o entregador alocado a cada chamada: `Chamada #i: entregador #j`, onde $i \in \{1, 2, \ldots k\}$ e $j \in \{1, 2, \ldots, n\}$, caso a pizza conste do cardápio de alguma filial do McHut. Em caso de empate na distância a ser percorrida, o entregador alocado deverá ser o de menor índice j. Caso contrário, você deve imprimir a mensagem `A McHut nao trabalha com este sabor`.

Exemplo

Entrada

```
4
0 0
1 3
1 2
4 3
2
0 4 3 quatroqueijos marguerita pizzaiolo
2 0 5 marguerita rucula bolonhesa portuguesa facciolina
6
3 2 portuguesa
0 2 marguerita
4 0 rucula
12 15 romana
7 8 rucula
9 10 marguerita
```

Saída

```
Chamada #1: entregador #1
Chamada #2: entregador #2
Chamada #3: entregador #1
Chamada #4: a McHut nao trabalha com este sabor
Chamada #5: entregador #1
Chamada #6: entregador #2
```

Problema 4: Etiqueta

```
Arquivo:  etiqueta.c ou etiqueta.pas
Entrada:  etiqueta.in
Saída:  etiqueta.out
```

Uma das primeiras tarefas em um interpretador de linguagem natural é fazer a etiquetagem das palavras, que consiste em identificar a classe morfológica e função sintática de cada uma das palavras na frase.
Neste problema vocês farão um etiquetador capaz de reconhecer frases simples da língua portuguesa. As frases que serão reconhecidas pelo seu etiquetador são formadas por:

```
<frase>   := [<sujeito>] <verbo> [<objeto>]
<sujeito> := [<artigo>] <substantivo>
<objeto>  := [<artigo>] <substantivo>
```

onde os colchetes indicam que a parte pode ou não ocorrer na frase. Assim, as frases abaixo poderão ser reconhecidas pelo seu etiquetador:

> O tricolor goleou o palmeiras
> Corinthians perdeu
> Esmagaram o santos

Os artigos permitidos são os artigos definidos (o, a, os, as), e os verbos são dados na entrada conforme aparecem nas frases (no exemplo acima, seriam dados os verbos "goleou", "perdeu" e "Esmagaram").
Você deve escrever um programa que recebe uma lista de verbos e em seguida uma lista de frases. Seu programa deve etiquetar as palavras de cada uma das frases conforme a classe morfológica e a função gramatical da palavra na frase. As frases serão dadas sem pontuação, estarão corretas (ou seja, são do tipo descrito acima) e verbos não poderão ser usados como substantivos (por exemplo a palavra "guarda" nas frases: *O guarda saiu* e *Joana guarda o casaco* teria as duas funções).

Entrada

A primeira linha do arquivo de entrada contém o número inteiro positivo k de verbos. A seguir, cada uma das k próximas linhas contém um verbo conforme aparece em uma das frases. As palavras da frase têm no máximo 31 caracteres, e as frases não terão mais de 127 caracteres. A linha seguinte contém o número n de frases. Cada uma das próximas n linhas contém uma frase.

Saída

Para cada frase você deverá imprimir o cabeçalho `Frase #i`, e nas linhas seguintes você deverá listar cada palavra da frase (na ordem em que aparece na frase) indicando a classe a que pertence (artigo, substantivo ou verbo), e, a seguir, deve indicar (quando houver) quem é o sujeito e objeto da frase.

Exemplo

Entrada

```
3
goleou
perdeu
Esmagaram
3
O tricolor             goleou     o palmeiras
Corinthians perdeu
Esmagaram o santos
```

Saída

```
Frase #1
O : artigo
tricolor : substantivo
goleou : verbo
o : artigo
palmeiras : substantivo
<sujeito> : O tricolor
<objeto> : o palmeiras

Frase #2
Corinthians : substantivo
perdeu : verbo
<sujeito> : corinthians

Frase #3
Esmagaram : verbo
o : artigo
santos : substantivo
<objeto> : o santos
```

Problema 5: Linhas Retilíneas

`Arquivo: retas.c ou retas.pas`
`Entrada: retas.in`
`Saída: retas.out`

Neste programa você deverá verificar as propriedades de uma curva desenhada no plano, formada apenas por segmentos de reta. A curva será desenhada com a ajuda de um *plotter*, que recebe uma seqüência de comandos no formato

```
D n
```

onde `D` é a direção em que o plotter deverá andar (pode ser N, S, E, W ou P – este último indica que o plotter deve parar) e n é o número de passos que o plotter deve dar naquela direção. Seu programa deverá reconhecer se a curva desenhada pelo plotter tem alguma das propriedades:

- *fechada* ou *aberta*: se no final do desenho da curva o plotter se encontra na mesma posição que estava no início dizemos que a curva é fechada.
- *simples* ou *complexa*: dizemos que uma curva é simples se o plotter toca o ponto de partida no máximo duas vezes, uma no início e outra quando o ploter pára sobre ele, e o plotter passa por qualquer outro ponto no máximo uma vez. Caso contrário dizemos que a curva é complexa.
- *convexa* ou *não-convexa*: Aplica-se apenas a curvas fechadas e simples. Dizemos que uma curva fechada é convexa se todos os ângulos internos são menores ou iguais a 180 graus. Caso contrário dizemos que a curva é não-convexa.

Entrada

A entrada é constituída por uma seqüência de linhas. Cada linha contém um comando ao plotter da forma `D n`, onde `D` indica a direção a ser percorrida (N, S, E, W ou P) e um inteiro positivo n indicando o número de passos que dever ser dado. A última linha contém `P 0`.

Saída

A saída deverá ser uma linha descrevendo as propriedades da curva na forma

```
A curva e: p1 p2 p3
```

onde `p1` pode ser `aberta` ou `fechada`, `p2` é `simples` ou `complexa` e, se for o caso, `p3` é `convexa` ou `nao-convexa`.

Exemplo 1

Entrada

```
S 9
E 9
N 9
W 9
P 0
```

Saída

```
A curva e: fechada simples convexa
```

Exemplo 2

Entrada

```
S 2
W 3
N 2
E 1
S 1
E 1
N 1
E 1
P 0
```

Saída

```
A curva e: fechada simples nao-convexa.
```

Exemplo 3

Entrada

```
E 1
S 1
W 2
S 1
E 1
N 2
P 0
```

Saída

```
A curva e: fechada complexa
```

Exemplo 4

Entrada

```
S 2
W 1
W 1
P 0
```

Saída

```
A curva e: aberta simples
```

Problema 6: Vinhos

```
Arquivo:  vinhos.c ou vinhos.pas
Entrada:  vinhos.in
Saída  :  vinhos.out
```

Um grande apreciador de vinhos planeja consumir uma garrafa de vinho de sua adega no final da tarde de cada sábado das próximas n semanas. Sua adega possui, infelizmente, somente n garrafas de vinho.
Sabe-se que a qualidade do vinho de uma garrafa i $(1 \leq i \leq n)$ pode ser medida em função da idade t_i do vinho (em semanas) através da fórmula $b_i t_i^2 - c_i t_i$, para constantes b_i e c_i que dependem do vinho. A idade atual do vinho da garrafa i, em semanas, é conhecida e denotada por a_i $(1 \leq i \leq n)$.
O apreciador deseja saber como ele deve consumir seu vinho para maximizar a qualidade total do vinho que ele vai consumir nas próximas n semanas (ou seja, a soma da qualidade de cada vinho ao ser consumido). Sua tarefa é fazer um programa que forneça ao apreciador a ordem em que os vinhos devem ser consumidos de forma a maximizar a qualidade total. Seu programa deve ser capaz de trabalhar com diversas instâncias em seqüência.

Entrada

A primeira linha de cada instância apresenta um inteiro positivo n, $n \leq 20$, representando o número de garrafas de vinho. Cada uma das próximas n linhas contêm os valores a_i, b_i e c_i, respectivamente, para $i = 1, \ldots, n$. O valor 0 (zero) para n indica o fim do arquivo de entrada.

Saída

Você deverá imprimir um cabeçalho indicando o número da instância que está tratando seguido da qualidade total (`Instancia #`k `-- Qualidade total:` q, onde k é o número da instância e q é a qualidade total da solução obtida). A próxima linha deve conter uma lista de números indicando a ordem em que as garrafas devem ser consumidas. Todos esses valores devem estar na mesma linha, separados por um espaço simples. Uma linha em branco deve ser deixada entre duas instâncias.

Exemplo

Entrada

```
6
10 2 10
5 3 15
4 5 20
20 3 5
10 1 25
15 4 15
0
```

Saída

```
Instancia #1 -- Qualidade total: 3050
5 2 1 3 6 4
```

Problema 7: Moedas

```
Arquivo:  moedas.c ou moedas.pas
Entrada:  moedas.in
Saída  :  moedas.out
```

Amadas por alguns e odiadas por muitos, as moedas estão em toda parte e fazem parte do cotidiano de qualquer pessoa. Isso acontece pois, afinal, guardadas as devidas proporções, elas representam dinheiro tanto quanto as notas.
Sua tarefa neste problema é elaborar um programa que ajude os diversos estabelecimentos comerciais a fornecer um valor de troco com o menor número de moedas possível. Visando a globalização da economia, seu programa deve ser capaz de trabalhar com moedas de diversos países. Um país por vez, no entanto.
Para que você possa realizar sua tarefa, serão fornecidos n inteiros positivos que representam os valores das moedas em circulação em um determinado país e uma coleção de m inteiros positivos que representam os valores para os quais você deverá calcular o equivalente em moedas. Sabe-se que o país com mais moedas no mundo é o Schredzistitão do Sul, que tem 100 moedas. Como dissemos anteriormente, sua resposta deve envolver o menor número possível de moedas.
Pode acontecer o caso de existir mais de uma configuração possível, isto é, pelo menos duas formas de combinar o mesmo número de moedas para atingir o valor desejado. Nesse caso, seu programa deve escolher a configuração que possua o maior número de moedas de mais alto valor.
Você deve observar também que, em alguns casos, dependendo dos valores das moedas e do troco desejado, pode ser impossível encontrar uma configuração adequada.

Entrada

Na primeira linha de cada instância é fornecido um inteiro n, $0 \leq n \leq 100$, de moedas com as quais seu programa deverá trabalhar. Um valor 0 (zero) para n indica o término do arquivo de entrada. Na próxima linha, caso $n \neq 0$, são fornecidos os n inteiros positivos n_i, com $1 \leq i \leq n$, que representam os valores das moedas com as quais você deverá realizar seus cálculos. Na terceira linha é fornecido o inteiro positivo m de valores para os quais o seu programa deverá calcular o equivalente em moedas. Na quarta linha são fornecidos estes m valores m_j, com $1 \leq j \leq m$, que são inteiros positivos menores ou iguais a 100.

Saída

Você deverá imprimir um cabeçalho indicando o número da instância que está tratando (`Instancia #`k, onde k é o número da instância). Nas próximas m linhas, devem ser apresentados os m valores, um por linha, na seguinte forma:

$$m_j\text{: } q_1,\ q_2,\ \ldots,\ q_n$$

onde q_i, com $1 \leq i \leq n$, representa o número de moedas de valor n_i utilizadas. Se, de acordo com os tipos das moedas utilizadas, um determinado valor m_j não puder ser expresso como combinação das mesmas, deve ser impressa a palavra `Impossivel` no lugar da lista de q_i's apresentada acima. Uma linha em branco deve ser deixada entre duas instâncias.

Exemplo

Entrada

```
6
1 5 10 25 50 100
4
3 5 30 95
4
2 4 6 8
2
5 12
3
25 20 5
1
40
0
```

Saída

```
Instancia #1
   3: 3 0 0 0 0 0
   5: 0 1 0 0 0 0
  30: 0 1 0 1 0 0
  95: 0 0 2 1 1 0

Instancia #2
   5: Impossivel
  12: 0 1 0 1

Instancia #3
  40: 0 2 0
```