Maratona de Programação
1998
Caderno de Problemas-Teste 1

Problema 1: Golfe

```
Arquivo: golfe.pas ou golfe.c
Entrada: golfe.in
Saída: golf.out
```

Todos que querem aprender a jogar golfe se deparam com diversas esquisitices (como qualquer jogo inventado pelos britânicos). Uma dessas esquisitices é o jeito de contar o número de tacadas que um jogador precisou para colocar a bola no buraco. Existe um “par” para cada buraco, e praticamente todos os resultados são relativos ao par (o par é o número de tacadas que um jogador razoável precisa para colocar a bola no buraco). Se o jogador precisar de uma tacada a menos que o par, dizemos que fez um “birdie”. Se foram duas a menos, ele fez um “eagle”, três a menos “double eagle”. Se ele precisar de uma a mais que o par, fez um “bogey”, duas a mais “double bogey”. Além disso, as tacadas espetaculares em que o jogador acerta a bolinha na primeira tacada recebem o nome de “hole-in-one”. Apesar de ser possível precisar de mais do que duas tacadas acima do par, e existirem nomes para isso, vamos assumir que o jogador trapaceará e dirá que fez um “double bogey”.
Você é o “caddy” do jogador (aquele que carrega os tacos e anota os resultados) e se confunde todo com esses nomes. Assim, você deve escrever um programa que dado o “par” para um buraco e o número de tacadas que o jogador precisou, imprime o nome da jogada que fez o jogador. Assegure-se de que você escreverá “hole-in-one” em uma jogada mesmo que ela tenha outro nome possível.

Entrada

Em cada linha do arquivo de entrada estão dois números inteiros (p e s), o primeiro com o “par” do buraco e o segundo com o número de tacadas que o jogador precisou, $3 \leq p \leq 5$ e $1 \leq s < 20$. Na última linha do arquivo de entrada teremos um buraco com $p = 0$ que não deverá causar nenhuma impressão na saída.

Saída

Para cada teste da entrada você deverá imprimir uma linha com o número do buraco (“`Buraco #1`”, “`Buraco #2`”, etc). Na linha de baixo deverá ter o nome da jogada obtida (“`Hole-in-one`”, “`Par`”, etc). Além disso deverá haver uma linha em branco depois de cada teste.

Exemplo

Entrada

```
5 3
3 1
4 7
```

```
0 0
```

Saída

```
Buraco #1
Eagle

Buraco #2
Hole-in-one

Buraco #3
Double Bogey
```

Problema 2: Grego

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

Na maior parte das línguas existentes no mundo os textos são escritos da esquerda para a direita. Entretanto, existem outras línguas em que os textos são lidos e escritos da direita para a esquerda. Como um primeiro passo de um tradutor automático, você deve escrever um programa que inverte cada frase da entrada, alinhando à direita para facilitar a visualização.

Entrada

O arquivo de entrada consiste de uma linha contendo o número máximo de caracteres em uma linha do arquivo de entrada, seguido por diversas linhas que deverão ser invertidas. O arquivo termina com um "end-of-file".

Saída

Para cada linha do arquivo está deverá ser impressa da direita para a esquerda, alinhando à direita (conforme o número máximo de caracteres por linha dado na entrada).

Exemplo

Entrada

```
70
Esta frase deve ficar muito esquisita vista de trás para frente.
A de baixo, entretanto, nem tanto:
Socorram-me subi no ônibus em Marrocos.
```

Saída

```
      .etnerf arap sárt ed atsiv atisiuqse otium racif eved esarf atsE
                                    :otnat men ,otnatertne ,oxiab de A
                               .socorram me subinô on ibus em-marrocoS
```

Problema 3: Compacto

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

Neste tempo da internet, conexões on line, email instantâneo, etc, ainda existem pessoas que precisam trabalhar com cópias em disquetes. A cada noite eles vão para casa e continuam a trabalhar em seus arquivos. Para copiar seus arquivos em disquete ele utiliza o seguinte procedimento:

1. Junta todos os arquivos em um grande arquivo SHAR.
2. Comprime o arquivo SHAR.
3. `uuencode` o arquio, que é um jeito de quebrá-lo em linhas bonitas de 62 caracteres cada (incluindo o fim-de-linha).
4. Divide o arquivo "uuencodificado" em arquivos menores com 30000 cada (que têm por volta de 1.86Mb).
5. Comprime cada arquivo e o coloca sozinho em um disquete.

O procedimento sempre funciona, já que 1.86Mb de texto uuencodificado, depois da compressão cabe com folga em um disquete 1.44Mb.
Com a compressão o tamanho do arquivo cai pela metade, e ao rodar o "uuencode" ele cresce de 50% (cada um arredondado para o inteiro mais próximo). Assim, dado o tamanho de um arquivo SHAR, queremos saber quantos disquetes precisaremos usar.

Entrada

Cada linha da entrada contém um número inteiro s,($0 \leq s \leq 1000000000$). A última linha contém um zero, que não deverá causar nenhuma saída.

Saída

Para cada linha, seu programa deverá imprimir uma linha com o número do teste ("`Teste #1`", "`Teste #2`", etc). Na linha seguinte você deverá imprimir quantos disquetes são necessários. Depois de cada teste deverá haver uma linha em branco.

Exemplo

Entrada

```
1000000
10000000
100000000
0
```

Saída

```
Teste #1
São necessários 1 disquetes.

Teste #2
São necessários 5 disquetes.

Teste #3
São necessários 41 disquetes.
```

Problema 4: Biblioteca

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

O grande problema que ocorre em bibliotecas é que os usuários dificilmente põem os livros de volta na estante no lugar correto. Com isso, uma prática bastante comum para evitar maiores problemas é pedir aos usuários que deixem os livros consultados sobre uma mesa, e então, um bibliotecário o colocará na posição correta. Nosso objetivo é tentar automatizar esta tarefa através de um programa.
Quando um usuário retira ou retorna um livro, o computador mantém um registro deste fato. Periodicamente os bibliotecários pedem ao programa quais os livros que foram devolvidos e estão na mesa para serem guardados. Os livros estão ordenados nas prateleiras pelo autor e (no caso de vários livros do mesmo autor) título. Para cada livro a ser devolvido à prateleira seu programa deve dar instruções sobre ao lado de qual obra o livro deve ser guardado na prateleira.

A entrada

Inicialmente é dado o acervo da biblioteca, com um livro em cada linha, sem qualquer ordem. Para cada livro é dado o nome dele entre aspas seguido pela palavra "por" e o autor do livro. No final do acervo ocorre uma linha com a palavra "END". Cada título e autor têm no máximo 80 caracteres e não contêm caracteres estranhos (como " por exemplo).
Depois do acervo segue uma série de registros de livros emprestados e devolvidos além de requisições de arrumações das prateleiras. Cada ordem ocorre em uma linha no seguinte formato:

> EMPRESTA "título"
> DEVOLVE "título"
> ARRUMA

A seqüência é terminada pela palavra "END".

A saída

Cada vez que a palavra "ARRUMA" aparece na entrada, os livros na mesa serão arrumados. Para isso uma seqüência de instruções são impressas na saída, uma por linha, no formato:

> Coloque "título 1" depois de "título 2"
> ou, no caso especial de ser o primeiro livro a ser guardado
> Coloque "título 1" na primeira posição

No final das instruções deve ocorrer a palavra "END" indicando ao bibliotecário que as instruções terminaram.

Exemplo

Entrada

```
"Meu pé direito" por Silva, J.
"Algoritmos intrincados" por Silva, J.
"A Linguagem de Programação C" por Kernighan, B. e Ritchie, D.
"Programação Sistemática" por Wirth, N.
END
EMPRESTA "Algoritmos intrincados"
EMPRESTA "A Linguagem de Programação C"
DEVOLVE "Algoritmos intrincados"
DEVOLVE "A Linguagem de Programação C"
ARRUMA
END
```

Saída

```
Coloque "Algoritmos intrincados" após "Meu pé direito"
Coloque "A Linguagem de Programação C" na primeira posição
```