CADERNO DE PROBLEMAS

10/05/2006 – 14:00 às 18:00

Leia atentamente estas instruções antes de iniciar a resolução dos problemas. Este caderno é composto de 5 problemas, 2 deles estão descritos em inglês.

1. É proibido utilizar livros, apostilas impressas, manuais ou qualquer outro material que não seja fornecido pela comissão organizadora durante a prova, com exceção de dicionários de inglês. É permitida a consulta do *help* do ambiente de programação se este estiver disponível e do material teórico referente às linguagens de programação (arquivos) gravados no D\ material.
2. A correção das resoluções dos problemas será semi-automatizada, baseada nos resultados obtidos com uma série de execuções dos algoritmos de cada equipe.
3. Siga atentamente as exigências da tarefa quanto ao formato da entrada e saída do seu algoritmo. Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
4. Os problemas não estão ordenados, neste caderno, por ordem de dificuldade. Procure resolver primeiro os problemas mais fáceis.
5. Utilize para nomear os seus arquivos-fonte os nomes indicados nos problemas de acordo com a linguagem de programação utilizada.
6. Cada equipe receberá 1 disquete com o nome da equipe que deve ser utilizado para a entrega dos algoritmos para correção.

Observações:

Não utilize arquivos para entrada ou saída. Todos os dados devem ser lidos da entrada padrão (teclado) e escritos na saída padrão (tela). Utilize as funções padrão para entrada e saída de dados:

- em Pascal: `readln`, `read`, `writeln`, `write`;
- em C: `scanf`, `getchar`, `printf`, `putchar`;
- em C++: as mesmas de C ou os objetos `cout` e `cin`;
- em JavaScript: `prompt`, `document.write`;

Quando a linguagem escolhida para a resolução dos problemas for o Pascal, recomendamos que não seja utilizada a instrução: `uses newdelay, crt;`

Faculdade de Informática de Presidente Prudente

http://www.unoeste.br/fipp/maratona

Monitor LCD (1-Vermelho)

Arquivo fonte: *monitor.c*, *monitor.cpp*, *monitor.pas* ou *monitor.html*

Um amigo seu comprou um computador novo há pouco. Antes disto, a máquina mais poderosa que ele já tinha usado foi uma calculadora eletrônica de bolso. Ele está um pouco desapontado porque ele gostava mais do monitor LCD da calculadora do que a tela do computador novo! Para torná-lo feliz, escreva um programa que exiba números no estilo LCD.

Entrada

A entrada contém várias linhas, uma para cada número a ser exibido. Cada linha contém os inteiros s e n, onde n é o número a ser exibido ($0 \leq n \leq 99.999.999$) e s é o tamanho dos dígitos no qual devem ser exibidos ($1 \leq s \leq 10$). A entrada terminará por uma linha que contém dois zeros que não deverão ser processados.

Exemplo de Entrada

```
2 12345
3 67890
0 0
```

Saída

Exiba os números especificados na entrada em um estilo de exibição LCD usando s caracteres "−" para os segmentos horizontais e s caracteres "|" para os verticais. Cada dígito ocupa exatamente $s + 2$ colunas e $2s + 3$ linhas. Preencha todo o espaço branco ocupado pelos dígitos com espaços em branco, inclusive o último dígito. Deve haver uma coluna de espaços em branco entre dois dígitos. Exiba uma linha em branco depois de cada número. Você encontrará um exemplo de cada dígito na amostra produzida a seguir.

Exemplo de Saída

```
    --   --         --
|     |     | |   | |
|     |     | |   | |
    --   --   --   --
| |        |     |     |
| |        |     |     |
    --   --         --

 ---   ---   ---   ---   ---
|          | |   | |   | |   |
|          | |   | |   | |   |
|          | |   | |   | |   |
 ---          ---   ---
|   |      | |   |      | |   |
|   |      | |   |      | |   |
|   |      | |   |      | |   |
 ---          ---   ---   ---
```

Criptografia (2-Verde Limão)

Arquivo fonte: *crypt.c*, *crypt.cpp*, *crypt.pas* ou *crypt.html*

Um método comum mas inseguro de criptografia de texto é permutar as letras do alfabeto. Em outras palavras, cada letra do alfabeto é substituída consistentemente no texto por alguma outra letra. Para assegurar que a criptografia é reversível, duas letras não são substituídas pela mesma letra.

Sua tarefa é decifrar várias linhas codificadas de texto, assumindo que cada linha usa um conjunto diferente de substituições, e que todas as palavras no texto decifrado são de um dicionário de palavras conhecidas.

Entrada

A entrada consiste de uma linha que contém um inteiro *n*, seguido por *n* palavras minúsculas, uma por linha, em ordem alfabética. Estas *n* palavras compõem o dicionário de palavras que podem aparecer no texto decifrado. Seguindo o dicionário são várias linhas de entrada. Cada linha é codificada como descrito acima.

Não existem mais que 1.000 palavras no dicionário. Nenhuma palavra excede 16 letras. As linhas codificadas contêm somente letras minúsculas e espaços e não excedem 80 caracteres de comprimento.

Exemplo de Entrada

```
6
and
dick
jane
puff
spot
yertle
bjvg xsb hxsn xsb qymm xsb rqat xsb pnetfn
xxxx yyy zzzz www yyyy aaa bbbb ccc dddddd
```

Saída

Decifre cada linha e exiba a saída padrão. Se não houver nenhuma solução, substitua todas as letras do alfabeto por um asterisco.

Exemplo de Saída

```
dick and jane and puff and spot and yertle
**** *** **** *** **** *** **** *** ******
```

Futebol (3-Verde Escuro)

Arquivo fonte: *futebol.c*, *futebol.cpp*, *futebol.pas* ou *futebol.html*

Futebol é o esporte mais popular no mundo. Um país no Mundo como o Brasil que foi cinco vezes campeão, tem muitos torneios nacionais e regionais. Sua tarefa é escrever um programa que receba o nome do torneio, nomes dos times e jogos realizados, e produzir como saída as posições do torneio.

Um time vence um jogo se marcar mais gols que seu oponente, e perde se marcar menos gols. Ambos os times pontuam se eles marcarem o mesmo número de gols. Um time ganha 3 pontos para cada jogo vencido, 1 ponto para cada empate, e 0 ponto para cada derrota.

Os times são classificados de acordo com estas regras (nesta ordem):
1. Mais pontos ganhos.
2. Mais vitórias.
3. Maior diferença de gols (isto é, gols marcados – gols contra)
4. Mais gols marcados.
5. Menor número de jogos.

Entrada

A primeira linha de entrada será apenas um inteiro N ($0 < N < 1.000$). Segue então N descrições de torneios, cada um começando com o nome do torneio. Estes nomes podem ser qualquer combinação de no máximo 100 letras, dígitos, espaços, etc., em uma única linha. A próxima linha conterá um número T ($1 < T \leq 30$), que representa o número de times que participam deste torneio. Na seqüência T linhas, cada uma contendo um nome de time. Nomes de time consistem em no máximo 30 caracteres, e podem conter qualquer caractere com código ASCII maior ou igual a 32 (espaço), com exceção dos caracteres "#" e "@".

Seguindo os nomes dos times, haverá um inteiro não negativo G em uma única linha que representa o número de jogos realizados neste torneio. G não será maior que 1.000. Após a linha com o inteiro G segue os resultados dos jogos realizados no formato:

Team_name_1#goals1@goals2#Team_name_2

Por exemplo, ***Team A#3@1#Team B*** significa que em um jogo entre ***Team A*** e ***Team B***, ***Team A*** marcou 3 gols e ***Team B*** marcou 1. Todos os gols serão inteiros não negativos menores que 20. Você pode assumir que todos os nomes dos times mencionados em resultados de jogos terão aparecido na seção de nomes de time, e que nenhum time jogará contra si mesmo.

Exemplo de Entrada

```
2
World Cup 1998 - Group A
4
Brazil
Norway
Morocco
Scotland
6
Brazil#2@1#Scotland
Norway#2@2#Morocco
Scotland#1@1#Norway
Brazil#3@0#Morocco
Morocco#3@0#Scotland
Brazil#1@2#Norway
Some strange tournament
5
Team A
Team B
Team C
Team D
Team E
5
Team A#1@1#Team B
Team A#2@2#Team C
Team A#0@0#Team D
Team E#2@1#Team C
Team E#1@2#Team D
```

Saída

Para cada torneio, você deve exibir o nome do torneio em uma única linha. Nas próximas *T* linhas você deve exibir as posições, de acordo com as regras acima. O formato de exibição para cada linha é mostrado abaixo:

[*a*]) *Team_name* [*b*]p, [*c*]g ([*d*]-[*e*]-[*f*]), [*g*]gd ([*h*]-[*i*])

onde [*a*] é a classificação do time, [*b*] é o total de pontos ganhos, [*c*] é o número de jogos realizados, [*d*] é o número de vitórias, [*e*] é o número de empates, [*f*] é o número de derrotas, [*g*] é a diferença de gols, [*h*] é o número de gols marcados, e [*i*] e o número de gols contra.

Deve haver um único espaço em branco entre os campos e uma única linha em branco entre os conjuntos de saída.

Exemplo de Saída

```
World Cup 1998 - Group A
1) Brazil 6p, 3g (2-0-1), 3gd (6-3)
2) Norway 5p, 3g (1-2-0), 1gd (5-4)
3) Morocco 4p, 3g (1-1-1), 0gd (5-5)
4) Scotland 1p, 3g (0-1-2), -4gd (2-6)

Some strange tournament
1) Team D 4p, 2g (1-1-0), 1gd (2-1)
2) Team E 3p, 2g (1-0-1), 0gd (3-3)
3) Team A 3p, 3g (0-3-0), 0gd (3-3)
4) Team B 1p, 1g (0-1-0), 0gd (1-1)
5) Team C 1p, 2g (0-1-1), -1gd (3-4)
```

Minesweeper (4-Laranja)

Arquivo fonte: *mine.c*, *mine.cpp*, *mine.pas* ou *mine.html*

Have you ever played Minesweeper? This cute little game comes with a certain operating system whose name we can't remember. The goal of the game is to find where all the mines are located within a $M \times N$ field.

The game shows a number in a square which tells you how many mines there are adjacent to that square. Each square has at most eight adjacent squares. The 4×4 field on the left contains two mines, each represented by a "*" character. If we represent the same field by the hint numbers described above, we end up with the field on the right:

```
*...        *100
....        2210
.*..        1*10
....        1110
```

Input

The input will consist of an arbitrary number of fields. The first line of each field contains two integers n and m ($0 < n, m \leq 100$) which stand for the number of lines and columns of the field, respectively. Each of the next n lines contains exactly m characters, representing the field.

Safe squares are denoted by "." and mine squares by "*," both without the quotes. The first field line where $n = m = 0$ represents the end of input and should not be processed.

Sample Input

```
4 4
*...
....
.*..
....
3 5
**...
.....
.*...
0 0
```

Output

For each field, print the message Field #*x*: on a line alone, where *x* stands for the number of the field starting from 1. The next *n*lines should contain the field with the "." characters replaced by the number of mines adjacent to that square. There must be an empty line between field outputs.

Sample Output

```
Field #1:
*100
2210
1*10
1110

Field #2:
**100
33200
1*100
```

WERTYU (5-Roxo)

Arquivo fonte: *wertyu.c*, *wertyu.cpp*, *wertyu.pas* ou *wertyu.html*

A common typing error is to place your hands on the keyboard one row to the right of the correct position. Then "**Q**" is typed as "**W**" and "**J**" is typed as "**K**" and so on. Your task is to decode a message typed in this manner.

Input

Input consists of several lines of text. Each line may contain digits, spaces, uppercase letters (except "**Q**", "**A**", "**Z**"), or punctuation shown above [except back-quote (`)]. Keys labeled with words [Tab, BackSp, Control, etc.] are not represented in the input.

Sample Input

```
O S, GOMR YPFSU/
```

Output

You are to replace each letter or punctuation symbol by the one immediately to its left on the QWERTY keyboard shown above. Spaces in the input should be echoed in the output.

Sample Output

```
I AM FINE TODAY.
```