OBI2004

CADERNO DE TAREFAS - MODALIDADE PROGRAMAÇÃO

22/5/2004 • 13:00 às 18:00

Leia atentamente estas instruções antes de iniciar a prova. Este caderno é composto de 13 páginas, numeradas de 1 a 13. Verifique se o caderno está completo.

1. É proibido consultar livros, anotações ou qualquer outro material durante a prova. É permitido a consulta ao *help* do ambiente de programação se este estiver disponível.
2. A correção é automatizada, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa.
3. Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
4. Todas as tarefas têm o mesmo valor na correção.
5. As tarefas não estão ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
6. Preste muita atenção no nome dos arquivos fonte indicados nas tarefas. Soluções na linguagem C devem ser arquivos com sufixo .c; soluções na linguagem C++ devem ser arquivos com sufixo .cc ou .cpp; soluções na linguagem Pascal devem ser arquivos com sufixo .pas.
7. Para problemas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada problema.
8. Ao final da prova, para cada solução que você queira submeter para correção, copie o arquivo fonte para o seu diretório de trabalho ou disquete, conforme especificado pelo seu professor.
9. Não utilize arquivos para entrada ou saída. Todos os dados devem ser lidos da entrada padrão (normalmente é o teclado) e escritos na saída padrão (normalmente é a 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*.
10. Procure resolver o problema de maneira eficiente. Na correção, eficiência também será levada em conta. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.

Sociedade Brasileira de Computação
http://www.sbc.org.br/olimpiada Email: sbc@sbc.org.br

Par ou ímpar

Arquivo fonte: par.c, par.cc, par.cpp ou par.pas

Muitas crianças gostam de decidir todas as disputas através do famoso jogo de Par ou Ímpar. Nesse jogo, um dos participantes escolhe Par e o outro Ímpar. Após a escolha, os dois jogadores mostram, simultaneamente, uma certa quantidade de dedos de uma das mãos. Se a soma dos dedos das mãos dos dois jogadores for par, vence o jogador que escolheu Par inicialmente, caso contrário vence o que escolheu Ímpar.

1. Tarefa

Dada uma seqüência de informações sobre partidas de Par ou Ímpar (nomes dos jogadores e números que os jogadores escolheram), você deve escrever um programa para indicar o vencedor de cada uma das partidas.

2. Entrada

A entrada é composta de vários conjuntos de testes. A primeira linha de um conjunto de testes contém um inteiro N, que indica o número de partidas de Par ou Ímpar que aconteceram. As duas linhas seguintes contêm cada uma um nome de jogador. Um nome de jogador é uma cadeia de no mínimo um e no máximo dez letras (maiúsculas e minúsculas), sem espaços em branco. As N linhas seguintes contêm cada uma dois inteiros A e B que representam o número de dedos que cada jogador mostrou em cada partida ($0 \leq A \leq 5$ e $0 \leq B \leq 5$). Em todas as partidas, o primeiro jogador sempre escolhe Par. O final da entrada é indicado por $N = 0$.

Exemplo de Entrada

```
3
Pedro
Paulo
2 4
3 5
1 0
2
Claudio
Carlos
1 5
2 3
0
```

3. Saída

Para cada conjunto de teste da entrada, seu programa deve produzir a saída da seguinte forma. A primeira linha deve conter um identificador do conjunto de teste, no formato “Teste *n*”, onde *n* é numerado seqüencialmente a partir de 1. As próximas N linhas devem indicar o nome do vencedor de cada partida. A próxima linha deve ser deixada em branco. A grafia mostrada no Exemplo de

Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
Pedro
Pedro
Paulo

Teste 2
Claudio
Carlos
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

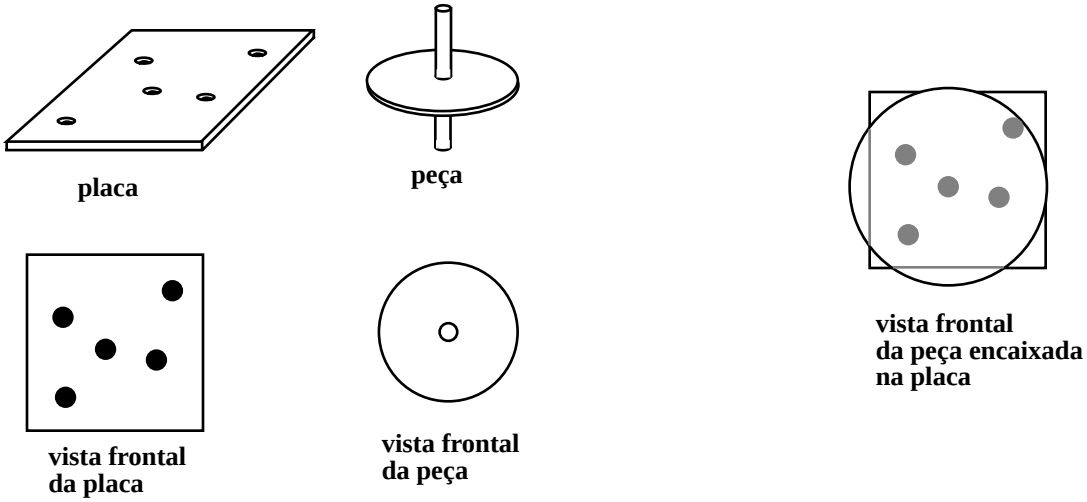
$0 \leq N \leq 1000$ ($N = 0$ apenas para indicar o fim da entrada)
$0 \leq A \leq 5$
$0 \leq B \leq 5$
$1 \leq$ *comprimento do nome de jogador* ≤ 10

Cubra os Furos

Arquivo fonte: furos.c, furos.cc, furos.cpp ou furos.pas

Uma placa de aço retangular contém *N* furos circulares de 5 mm de diâmetro, localizados em pontos distintos, não sobrepostos – ou seja, o centro de cada furo está a uma distância maior ou igual a 5 mm do centro de todos os outros furos.

Uma peça de forma circular, tendo em seu centro um eixo de 5 mm de diâmetro, deve ser colocada sobre a placa, de modo que o eixo encaixe-se em um de seus furos.



1. Tarefa

Você deve escrever um programa para determinar o diâmetro mínimo que a peça deve ter de tal forma que, com seu eixo encaixado em um dos furos da placa, a parte circular cubra completamente todos os outros furos da placa.

2. Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de teste contém um inteiro *N*, que indica o número de furos na placa de aço ($1 \leq N \leq 1000$). As *N* linhas seguintes contêm cada uma dois inteiros *X* e *Y*, separados por um espaço em branco, que descrevem a posição do centro de um furo ($-10000 \leq X \leq 10000$ e $-10000 \leq Y \leq 10000$). A unidade de medida das coordenadas dos furos é 1 mm. O final da entrada é indicado por $N = 0$.

Exemplo de Entrada

```
3
20 25
10 5
10 10
3
```

```
0 5
10 0
0 10
0
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve produzir três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato "Teste *n*", onde *n* é numerado seqüencialmente a partir de 1. A segunda linha deve conter o diâmetro mínimo que a peça deve ter, como um número inteiro. A terceira linha em deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
41

Teste 2
27
```

(esta saída corresponde ao exemplo de entrada acima)

ERRATA:

Exemplo de Saída

```
Teste 1
42

Teste 2
28
```

Na correção, as várias possibilidades de arredondamento foram consideradas corretas.

4. Restrições

$0 \leq N \leq 1000$ ($N = 0$ apenas para indicar o fim da entrada)
$-10000 \leq X \leq 10000$
$-10000 \leq Y \leq 10000$

Palíndrome

Arquivo fonte: pal.c, pal.cc, pal.cpp ou pal.pas

Uma cadeia de caracteres é chamada de *palíndrome* se seqüência de caracteres da esquerda para a direita é igual à seqüência de caracteres da direita para a esquerda (uma outra definição é que o primeiro caractere da cadeia deve ser igual ao último caractere, o segundo caractere seja igual ao penúltimo caractere, o terceiro caractere seja igual ao antepenúltimo caractere, e assim por diante). Por exemplo, as cadeias de caracteres ‘mim’, ‘axxa’ e ‘ananaganana’ são exemplos de palíndromes.

Se uma cadeia não é palíndrome, ela pode ser dividida em cadeias menores que são palíndromes. Por exemplo, a cadeia ‘aaxyx’ pode ser dividida de quatro maneiras distintas, todas elas contendo apenas cadeias palíndromes: {‘aa’, ‘xyx’}, {‘aa’, ‘x’, ‘y’, ‘x’}, {‘a’, ‘a’, ‘xyx’} e {‘a’, ‘a’, ‘x’, ‘y’, ‘x’}.

1. Tarefa

Escreva um programa que determine qual o menor número de partes em que uma cadeia deve ser dividida de forma que todas as partes sejam palíndromes.

2. Entrada

A entrada é constituída de vários conjuntos de teste. A primeira linha de um conjunto de testes contém um inteiro N que indica o número de caracteres da cadeia ($1 \leq N \leq 2000$). A segunda linha contém a cadeia de caracteres, composta por letras minúsculas (de ‘a’ a ‘z’), sem espaços em branco. O final da entrada é indicado por $N = 0$.

Exemplo de Entrada

```
3
axa
6
xyzyyx
10
bbabcbbaab
0
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve produzir três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato “Teste *n*”, onde *n* é numerado a partir de 1. A segunda linha deve conter um inteiro indicando o menor número de partes que a cadeia de entrada deve ser dividida de forma que todas as partes sejam palíndromes. A terceira linha deve ser deixada em branco. O formato mostrado no exemplo de saída abaixo deve ser seguido rigorosamente.

Exemplo de Saída

```
Teste 1
1

Teste 2
4

Teste 3
4
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq N \leq 2000$ ($N = 0$ apenas para indicar o fim da entrada)

TV da Vovó

Arquivo fonte: tv.c, tv.cc, tv.cpp ou tv.pas

A vovó tem um televisor muito antigo, que ultimamente está exibindo um defeito incômodo: a imagem aparece ‘deslocada’ (para cima ou para baixo, para o lado direito ou para o lado esquerdo). Quando a imagem está deslocada para cima, a parte da imagem que deixa de ser vista na parte superior reaparece na parte de baixo da tela. Da mesma forma, quando a imagem está deslocada a direita, a parte da imagem que deixa de ser vista à direita reaparece na tela do lado esquerdo.

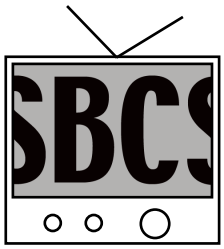


imagem deslocada para a esquerda

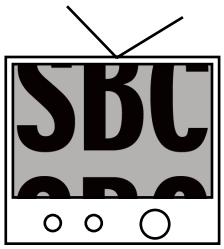


imagem deslocada para cima

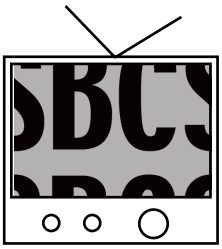


imagem deslocada para a esquerda e para cima

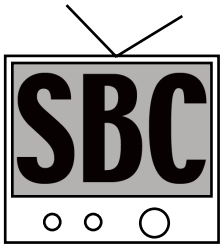


imagem corrigida

A imagem do televisor pode ser vista como uma matriz de pontos organizados em linhas e colunas. Para consertar o televisor da vovó, você pode ajustar a imagem introduzindo uma série de ‘comandos de correção’ em um painel de ajuste. Cada comando de correção desloca a imagem de um certo número de linhas (para cima ou para baixo) e um certo número de colunas (para a direita ou para a esquerda).

1. Tarefa

Dada uma matriz que representa uma imagem defeituosa e uma série de comandos de correção, seu programa deve calcular a matriz que representa a imagem resultante após todos os comandos terem sido aplicados sequencialmente.

2. Entrada

A entrada possui vários conjuntos de teste. Cada conjunto de teste inicia com a descrição da matriz que representa a imagem do televisor. A primeira linha contém dois inteiros M e N representando o número de linhas e o número de colunas da matriz ($1 \leq M \leq 1000$ e $1 \leq N \leq 1000$). As M linhas seguintes da entrada contém cada uma N inteiros, descrevendo o valor de cada ponto da imagem. Após a descrição da imagem, segue-se a descrição dos comandos de correção. Cada comando de correção é descrito em uma linha contendo dois inteiros X e Y. O valor de X representa o deslocamento na direção horizontal (valor positivo representa deslocamento para a direita, valor negativo para a esquerda), e o valor de Y representa o deslocamento da direção vertical (valor positivo para cima, valor negativo para baixo). O final da lista de comandos é indicado por $X = Y = 0$, e o final da entrada é indicado por $M = N = 0$.

Exemplo de Entrada

```
3 3
1 2 3
4 5 6
7 8 9
1 0
1 -1
0 0
3 4
6 7 8 5
10 11 12 9
2 3 4 1
-3 2
0 0
0 0
```

3. Saída

Para cada conjunto de teste, o seu programa deve produzir uma imagem na saída. A primeira linha da saída deve conter um identificador do conjunto de teste, no formato "Teste *n*", onde *n* é numerado seqüencialmente a partir de 1. A seguir deve aparecer a matriz que representa a imagem resultante, no mesmo formato da imagem de entrada. Ou seja, as *N* linhas seguintes devem conter cada uma *M* inteiros que representam os pixels da imagem. Após a imagem deixe uma linha em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
8 9 7
2 3 1
5 6 4

Teste 2
1 2 3 4
5 6 7 8
9 10 11 12
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restricões

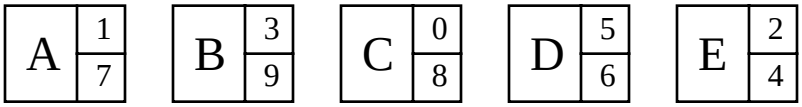
$0 \leq N \leq 1000$ ($N = 0$ apenas para indicar o final da entrada)
$0 \leq M \leq 1000$ ($M = 0$ apenas para indicar o final da entrada)
$0 \leq X \leq 1000$
$0 \leq Y \leq 1000$
$0 \leq$ *número de comandos de correção em cada conjunto de teste* ≤ 1000

Proteja sua senha

Arquivo fonte: senha.c, senha.cc, senha.cpp ou senha.pas

Por questões de segurança, muitos bancos hoje em dia estão alterando a forma como seus clientes digitam as senhas nos caixas eletrônicos, pois alguém pode postar-se atrás do cliente e ver as teclas à medida em que ele as digita.

Uma alternativa bastante utilizada tem sido associar os dez dígitos a cinco letras, de forma que cada letra esteja associada a dois dígitos, conforme o exemplo abaixo:



As associações entre números e letras são mostradas como botões numa tela sensível ao toque, permitindo que o cliente selecione os botões correspondentes à senha. Considerando a disposição dos botões da figura acima, a senha 384729 seria digitada como BCEAEB (note que a mesma seqüência de letras seria digitada para outras senhas, como por exemplo 982123).

Cada vez que o cliente usa o caixa eletrônico, as letras utilizadas são as mesmas (de ‘A’ a ‘E’), com os botões nas mesmas posições, mas os dígitos são trocados de lugar. Assim, caso um intruso veja (mesmo que mais de uma vez) a seqüência de letras digitada, não é possível notar facilmente qual a senha do cliente do banco.

1. Tarefa

Dada uma seqüência de associações entre letras e números, e as letras digitadas pelo cliente do banco para cada uma dessas associações, você deve escrever um programa para determinar qual é a senha do cliente.

2. Entrada

A entrada é composta de vários conjuntos de testes. A primeira linha de um conjunto de testes contém um inteiro N, que indica o número de associações entre letras e números e as senhas digitadas ($2 \leq N \leq 10$). As N linhas seguintes contêm as entradas da seguinte forma: 10 dígitos, em ordem de associação, para as letras de ‘A’ a ‘E’ (2 dígitos para a letra A, 2 para a B e assim sucessivamente) e 6 letras que representam a senha codificada conforme os dígitos anteriores. As N associações fornecidas em um conjunto de testes serão sempre suficientes para definir univocamente a senha do cliente. O final da entrada é indicado por $N = 0$.

Exemplo de Entrada

```
2
1 7 3 9 0 8 5 6 2 4 B C E A E B
9 0 7 5 8 4 6 2 3 1 E C C B D A
3
```

```
0 1 2 3 4 5 6 7 8 9 B C D D E E
1 3 5 4 6 8 7 9 0 2 E B C D C D
3 2 0 4 5 9 7 6 8 1 A C D D E C
0
```

3. Saída

Para cada conjunto de teste da entrada, seu programa deve produzir três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato "Teste *n*", onde *n* é numerado seqüencialmente a partir de 1. A segunda linha deve conter a senha do cliente, com um espaço após cada dígito. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
3 8 4 7 2 9

Teste 2
2 5 6 7 8 9
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$2 \leq N \leq 10$ ($N = 0$ apenas para indicar o fim da entrada)

Orkut

Arquivo fonte: orkut.c, orkut.cc, orkut.cpp ou orkut.pas

Larissa acaba de entrar para o Orkut, um site na internet que permite que as pessoas se reúnam em comunidades e grupos de amigos. Como ela acabou de se registrar, ela ainda não possui muitos amigos na sua lista de contatos. Após fazer uma pesquisa, ela descobriu que os seus antigos amigos de escola (que adoravam mexer com computadores) também fazem parte do Orkut. Larissa então decidiu chamá-los para serem seus amigos virtuais. Porém, eles resolveram brincar com a Larissa, e cada um deles só vai aceitar o pedido de Larissa quando ela já for amiga virtual de alguns dos outros amigos do grupo. Assim, para conseguir ter todos os seus antigos amigos de escola na sua lista de amigos do Orkut, ela deve cumprir as exigências de cada um deles.

1. Tarefa

Larissa acha que pode encontrar uma seqüência de nomes dos amigos, de modo que se ela pedir a cada um deles para ser sua amiga no Orkut, obedecendo a seqüência, todas as exigências serão cumpridas e todos eles irão aceitar o seu pedido. Larissa precisa da sua ajuda para resolver esse problema de forma rápida. A sua tarefa é escrever um programa para encontrar uma seqüência de nomes que resolva o problema, ou dizer que não é possível encontrar tal seqüência.

2. Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de teste contém um inteiro N que indica o número de antigos amigos da Larissa ($1 \leq N \leq 30$). A linha seguinte irá conter N nomes de amigos, separados por espaço em branco. Cada nome não terá mais de 15 letras, e serão todos distintos. Nas próximas N linhas serão indicadas as exigências que a Larissa deve cumprir. Cada linha descreve a exigência de um amigo e começará com o nome desse amigo, seguido de um número M ($0 \leq M \leq N–1$), que indica o número de pessoas que aquele amigo quer que a Larissa seja amiga antes, e seguido pelos M nomes de amigos (cada item na linha separado por espaço em branco). O final da entrada é indicado por $N = 0$.

Exemplo de Entrada

```
5
Joao Maria Tadeu Jose Ricardo
Joao 2 Maria Ricardo
Maria 1 Tadeu
Jose 1 Joao
Tadeu 0
Ricardo 0
3
Joao Maria Renata
Maria 1 Joao
Joao 1 Renata
Renata 1 Maria
0
```

3. Saída

Para cada conjunto de teste seu programa deve produzir três linhas na saída. A primeira linha deverá conter um identificador do conjunto de teste, no formato “Teste *n*”, onde *n* é numerado seqüencialmente a partir de 1. A segunda linha deve conter a seqüência de nomes de amigos (cada nome seguido de um espaço em branco) que resolve o problema da Larissa, ou a palavra “impossivel”, quando não houver uma seqüência possível (note a ausência de acentuação). Se existir mais de uma seqüência de amigos que resolve o problema, imprima qualquer uma delas (mas apenas uma). A terceira linha deverá ser deixada em branco. A grafia mostrada no Exemplo de Saída abaixo deverá ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
Ricardo Tadeu Maria Joao Jose

Teste 2
impossivel
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq N \leq 30$ ($N = 0$ apenas para indicar o fim da entrada)
$0 \leq M \leq N - 1$
Cada nome de amigo terá no máximo 15 letras