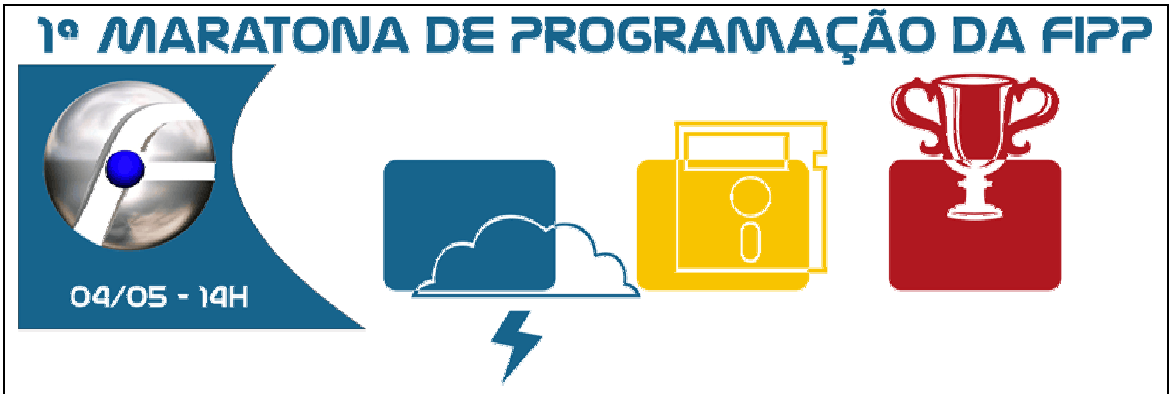


CADERNO DE PROBLEMAS

04/05/2005 – 14:00 às 18:00

Leia atentamente estas instruções antes de iniciar a resolução dos problemas. Este caderno é composto de 5 problemas.

1. É proibido utilizar livros, apostilas impressas, manuais ou qualquer outro material que não seja fornecido pela comissão organizadora durante a prova. É permitida a consulta do *help* do ambiente de programação se este estiver disponível e do material teórico referente às linguagens se 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`;

Faculdade de Informática de Presidente Prudente

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

Complemento de Senha de e-mail (1)

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

Para reforçar ainda mais a segurança da senha das contas de e-mail, a FIPP resolveu modificar o esquema de segurança baseado em criptografia de senha. Foi proposto um complemento além da senha para cada usuário. A forma de como os usuários digitarão este complemento alterna toda vez que eles acessam o software para leitura e ou envio de e-mails.

A alternativa escolhida para isto foi associar os dez dígitos numéricos a cinco letras, de forma que cada letra esteja associada a dois dígitos, conforme o exemplo abaixo:

V	1	7
W	3	9
X	0	8
Y	5	6
Z	2	4

As associações entre números e letras são mostradas na tela para o usuário, permitindo que este selecione as letras que corresponde ao complemento da senha. Considerando o exemplo acima, o complemento 384729 seria digitado como WXZVZW (note que a mesma seqüência de letras seria digitada para outras senhas, como por exemplo 982123).

Cada vez que o usuário acessa o software para leitura ou envio de e-mails, as letras utilizadas são as mesmas (de 'V' a 'Z'), mas as associações dos dígitos são mudadas. Dessa forma, caso algum intruso veja (mesmo que mais de uma vez) a seqüência de letras digitadas, não é possível notar facilmente qual é o complemento da senha do usuário da conta de e-mail.

1. Tarefa

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

2. Entrada

A entrada é composta de vários conjuntos de teste. 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 'V' a 'Z' (2 dígitos para a letra V, 2 para a W 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 W X Z V Z W
9 0 7 5 8 4 6 2 3 1 ZX X W Y V
3
0 1 2 3 4 5 6 7 8 9 W X Y Y Z Z
1 3 5 4 6 8 7 9 0 2 Z W X Y X Y
3 2 0 4 5 9 7 6 8 1 V X Y Y Z X
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)

Palíndrome (2)

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

Uma cadeia de caracteres é chamada de *palíndrome* se a 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 divida 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 de 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 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 de entrada)

Staleka (3)

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

Na última edição do programa de televisão Big Brother da Rede Globo, foi criada uma moeda corrente chamada de "*Staleka*" que os participantes usavam para comprar alimentos, materiais de limpeza, toalhas, etc. Existem notas de St$ 100,00, St$ 50,00, St$10,00, St$5,00 e St$1,00. Você foi contratado(a) para ajudar na programação do caixa automático instalado dentro da casa do Big Brother.

1. Tarefa

O caixa eletrônico da casa do Big Brother opera com todos os tipos de notas disponíveis, mantendo um estoque de cédulas para cada valor (St$100,00, St$ 50,00, St$10,00, St$5,00 e St$1,00). Os participantes da casa utilizam o caixa eletrônico para efetuar retiradas de um certo número inteiro de Stalekas.

Sua tarefa é escrever um programa que, dado o valor de Stalekas desejado pelo participante da casa, determine o número de cada uma das notas necessário para totalizar esse valor, de modo a minimizar a quantidade de cédulas entregues. Por exemplo, se o participante deseja retirar St$50,00, basta entregar uma única nota de cinqüenta Stalekas. Se o participante deseja retirar St$72,00, é necessário entregar uma nota de St$50,00, duas de St$10,00 e duas de St$1,00.

2. Entrada

A entrada é composta de vários conjuntos de teste. Cada conjunto de teste é composto por uma única linha, que contém um número inteiro positivo V, que indica o valor solicitado pelo participante da casa. O final da entrada é indicado por $V = 0$.

Exemplo de Entrada

```
1
72
0
```

3. Saída

Para cada conjunto de teste de 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. Na segunda linha devem aparecer cinco inteiros *I*, *J*, *K, L e M* que representam o resultado encontrado pelo seu programa: *I* indica o número de cédulas de St$100,00, *J* indica o número de St$50,00, *K* indica o número de cédulas de St$10,00, *L* indica o número de cédulas de St$5,00 e *M* indica o número de cédulas de St$1,00. 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
0 0 0 0 1

Teste 2
01 2 0 2
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

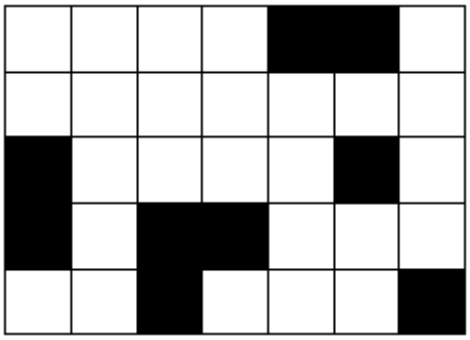
$0 \leq V \leq 10000$ ($V = 0$ apenas para indicar o fim da entrada)

Palavras Cruzadas (4)

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

O conhecido passatempo de palavras cruzadas é composto por uma grade retangular de quadrados brancos e pretos e duas listas de definições. Uma das listas de definições é para palavras escritas da esquerda para a direita nos quadrados brancos (nas linhas) e a outra lista é para palavras que devem ser escritas de cima para baixo nos quadrados brancos (nas colunas). Uma palavra é uma seqüência de *dois ou mais* caracteres do alfabeto. Para resolver um jogo de palavras cruzadas, as palavras correspondentes às definições devem ser escritas nos quadrados brancos da grade.

A figura abaixo ilustra um exemplo de jogo de palavras cruzadas.



Um exemplo de jogo de palavras cruzadas

Uma palavra *horizontal* é escrita em uma seqüência de quadrados brancos em uma linha, iniciando-se em um quadrado que tem um quadrado preto à esquerda ou que está na primeira coluna à esquerda. A seqüência de quadrados para essa palavra continua da esquerda para a direita, terminando no quadrado branco imediatamente anterior a um quadrado preto, ou no quadrado branco da coluna mais à direita da grade.

Uma palavra *vertical* é escrita em uma seqüência de quadrados brancos em uma coluna, iniciando-se em um quadrado que tem um quadrado preto acima ou que está na primeira linha acima. A seqüência de quadrados para essa palavra continua de cima para baixo, terminando no quadrado branco imediatamente anterior a um quadrado preto, ou no quadrado branco da coluna mais abaixo da grade.

1. Tarefa

Sua tarefa é escrever um programa que recebe como entrada vários jogos de palavras cruzadas *resolvidas* e produzir as listas de palavras verticais e horizontais que constituem as soluções.

2. Entrada de Dados

A primeira linha de um conjunto de testes contém dois inteiros positivos, M e N, que indicam respectivamente o número de linhas e o número de colunas do jogo de palavras cruzadas. Cada uma das *M* linhas seguintes contém *N* caracteres (caracteres do alfabeto ou o caractere '*'), correspondendo a um jogo de palavras cruzadas resolvido. O caractere '*' é utilizado para representar um quadrado preto. O final do conjunto de

testes é indicado quando M = N = 0 (este último conjunto de testes não é válido e não deve ser processado).

Exemplo de Entrada

```
1 8
*PASCAL*
3 3
*M*
BIT
*L*
5 7
ATOS**J
MEMORIA
*COLE*V
*L**DIA
LA*VER*
0 0
```

3. Saída de Dados

Para cada conjunto de teste seu programa deve produzir duas listas, uma para as palavras horizontais e uma para as palavras verticais. A lista das palavras horizontais deve ser precedida por uma linha de cabeçalho onde está escrito “Horizontais:”; este cabeçalho só deve aparecer quando houver palavras horizontais no jogo (por exemplo, em um jogo com apenas uma coluna não há palavras horizontais). Da mesma forma, a lista das palavras verticais deve ser precedida por uma linha onde está escrito “Verticais:”. As listas devem ser numeradas e apresentadas na ordem crescente de numeração da grade original. Deve ser deixada uma linha em branco após cada teste. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
Horizontais:
PASCAL

Teste 2
Horizontais:
BIT
Verticais:
MIL

Teste 3
Horizontais:
ATOS
MEMORIA
COLE
DIA
LA
VER
Verticais:
AM
TECLA
OMO
SOL
REDE
IR
JAVA
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$1 \leq M \leq 100$
$1 \leq N \leq 100$
$M = 0$ e $N = 0$ apenas para indicar o fim dos testes.

Romanos (5)

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

1. Tarefa

Os romanos usavam um sistema interessante para representar os números. Eles usavam sete letras do alfabeto e a cada uma delas atribuíam valores:

I	V	X	L	C	D	M
1	5	10	50	100	500	1.000

O numerais I, X, C, M só podem ser repetidos até três vezes.

I = 1	II = 2	III = 3
X = 10	XX = 20	XXX = 30
C = 100	CC = 200	CCC = 300
M = 1000	MM = 2000	MMM = 3000

A tabela abaixo mostra algumas representações de numerais romanos.

I = 1	XX = 20	CCC = 300
II = 2	XXX = 30	CD = 400
III = 3	XL = 40	D = 500
IV = 4	L = 50	DC = 600
V = 5	LX = 60	DCC = 700
VI = 6	LXX = 70	DCCC = 800
VII = 7	LXXX = 80	CM = 900
VIII = 8	XC = 90	M = 1.000
IX = 9	C = 100	MM = 2.000
X = 10	CC = 200	MMM = 3.000

Os numerais I, X e C, escritos à direita de numerais maiores, somam-se seus valores aos desses numerais.

Exemplos:

VII = 7 (5 + 2)	LX = 60 (50 + 10)	LXXIII = 73 (50+ 20 + 3)
CX = 110 (100 + 10)	CXXX = 130 (100 + 30)	MCC = 1200 (1000 + 200)

Os numerais I, X e C, escritos à esquerda de numerais maiores, subtraem-se seus valores aos desses numerais.

Exemplos:

IV = 4 (5 – 1)	IX = 9 (10 – 1)	XL = 40 (50 – 10)
XC = 90 (100 – 10)	CD = 400 (500 – 100)	CM = 900 (1000 – 100)

2. Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de testes contém um número inteiro N que indica quantos dígitos representam o número para ser convertido em romanos. A linha seguinte contém, o número a ser convertido em romanos. O final da entrada é indicado por $N = 0$.

Exemplo de Entrada

```
2
55
4
1256
3
987
0
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve produzir duas 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 a representação em romanos do número inteiro convertido. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
LV

Teste 2
MCCLVI

Teste 3
CMLXXXVII
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq N \leq 6$ ($N = 0$ apenas para indicar o final da entrada)