Universidade Federal do Espírito Santo
Departamento de Informática
Programa de Educação Tutorial – PET EngComp
E-mail: petengcomp@inf.ufes.br
Home-Page: www.inf.ufes.br/~pet
Tel. (27) 3335-2161

Topcom 6

Torneio de Programação de Computadores

PROVA

Realização:

Apoio:

Patrocínio:

PROBLEMA A
NÚMERO DE ERDÖS

O Húngaro Paul Erdös (1913-1996) não foi somente um dos mais estranhos matemáticos do século 20, ele também esteve entre os mais famosos. Ele continuou publicando artigos até uma idade avançada e todo matemático que teve a honra de ser seu co-autor é muito respeitado.

Nem todo mundo teve a chance de ser co-autor de Erdös, logo muita gente ficava satisfeita se conseguisse publicar um artigo com alguém que já publicou com Erdös. Isso criou os números de Erdös. Quem foi co-autor de Erdös tem o número 1 de Erdös. Um autor que não tenha publicado com Erdös, mas com alguém que tenha o número 1 de Erdös obtém o número 2 de Erdös, e assim por diante. Hoje em dia, quase todo mundo quer saber qual número de Erdös possui. Sua tarefa é escrever um programa que calcule o número de Erdös de um grupo de cientistas.

A ENTRADA

A entrada possui uma seqüência de instâncias, cada instância consistindo de uma base de dados de artigos e uma lista de nomes. Uma instância começa com a linha "p n", onde p e n são números naturais com 1<= p < = 32000, 1<= n <= 3000. Em seguida, p linhas contendo a descrição dos artigos (essa é a base de dados de artigos). Um artigo é descrito por uma linha da seguinte forma:

ÚltimoNome1, PrimeiroNome1, ÚltimoNome2, PrimeiroNome2, ...: Título

Os nomes e o título podem conter quaisquer caracteres ASCII entre 32 e 126, exceto vírgulas e dois-pontos. Sempre haverá exatamente um espaço em branco depois de uma vírgula. O primeiro nome poderá vir abreviado, mas o mesmo nome sempre virá escrito da mesma forma. Em particular, o nome de Erdös sempre será escrito como "Erdos, P.". (Acentos não são considerados).

Depois de p artigos, seguem n linhas cada uma contendo exatamente um nome no mesmo formato do banco de dados de artigos.

A linha "0 0" encerra a entrada.

Nenhum nome consistirá de mais de 40 caracteres. Nenhuma linha contém mais de 250 caracteres. Em cada instância haverá no máximo 10000 autores diferentes.

A SAÍDA

Para cada instância, primeiro imprima o número da instância no formato mostrado. Então imprima para cada autor na lista de nome, o seu número de Erdös baseado nos artigos da base de dados da instância. A ordem dos autores deverá seguir a ordem de entrada. Autores que não possuem relação com Erdös via artigos, possuem número de Erdös infinity. Para

autores que possam ter dois números de Erdös, imprima o menor deles. Atenha-se ao formato mostrado no exemplo de saída.

Depois de cada conjunto de banco de dados, deve haver uma linha em branco.

EXEMPLO DE ENTRADA

```
2 2
Smith, M.N., Martin, G., Erdos, P.: Newtonian forms of prime factors matrices
Gardner, M., Martin, G.: Commuting Names
Smith, M.N.
Gardner, M.
0 0
```

EXEMPLO DE SAÍDA

```
Database #1
Smith, M.N.: 1
Gardner, M.: 2
```

PROBLEMA B
(4)009-2161

Empresários gostam de ter números de telefone fáceis de serem memorizados. Uma maneira de fazer isso é ter um número que signifique uma palavra ou frase. No nosso problema, os números só possuem 7 dígitos. Então, desconsidere o primeiro. Dessa forma, você pode ligar para o Cine Metrópolis discando DEL-AERO. Às vezes, somente parte do número é usado para soletrar uma palavra. Supondo que depois do Torneio, você queira pedir uma pizza da Pizzaria do Gino, você poderia discar 310-GINO. Uma outra maneira de tornar um número fácil de ser gravado é agrupando os dígitos. Você também poderia pedir uma pizza discando "três dez" número 3-10-10-10.

Considere que a forma padrão de um número de telefone é sete dígitos decimais com um hífen entre o terceiro e o quarto dígitos (ex: 777-5321). O teclado de um telefone possui mapeamento de letras e números, da seguinte forma:

A, B e C mapeiam 2
D, E e F mapeiam 3
G, H e I mapeiam 4
J, K e L mapeiam 5
M, N e O mapeiam 6
P, R e S mapeiam 7
T, U e V mapeiam 8
W, X e Y mapeiam 9

Não há mapeamento para Q ou Z. Hifens não são discados e podem ser removidos se necessário. A forma padrão de DEL-AERO é 335-2376, a de 310-GINO é 310-4466 e a de 3-10-10-10 é 310-1010.

Dois números de telefone são equivalente se têm a mesma forma padrão (Disca-se o mesmo número.)

Sua companhia está compilando um diretório de telefones de empresários locais. Como parte do controle de qualidade do processo, você quer checar que dois ou mais empresários não têm o mesmo número no diretório.

A ENTRADA

A entrada consiste de um único caso. A primeira linha especifica a quantidade de números de telefones no diretório (até 100.000) como um inteiro positivo sozinho em uma linha. Cada telefone consiste de uma string de números decimais, letras maiúsculas (exceto Q e Z) e hifens. Exatamente sete dos caracteres serão letras ou números.

A SAÍDA

Você deve gerar uma linha para cada número de telefone que apareça mais de uma vez no

diretório em qualquer forma. Essa linha deverá ter o telefone na sua forma padrão, seguido de um espaço, seguido da quantidade de vezes que ele aparece no diretório. As linhas da saída deverão vir em ordem lexicográfica crescente. Se não há repetições na entrada imprima:

No duplicates.

EXEMPLO DE ENTRADA

```
12
4873279
ITS-EASY
888-4567
3-10-10-10
888-GLOP
TUT-GLOP
967-11-11
310-GINO
F101010
888-1200
-4-8-7-3-2-7-9-
487-3279
```

EXEMPLO DE SAÍDA

```
310-1010 2
487-3279 4
888-4567 3
```

PROBLEMA C
COMO POSSO VOLTAR PARA CASA?

Este problema é baseado em http://acm.uva.es/p/ e adaptado pela Prof.ª Dr.ª Patrícia Dockhorn Costa, do Departamento de Informática da UFES.

O objetivo desta questão é gerar instruções de retorno de uma determinada localização baseado nas instruções de ida. Por exemplo, imagine as seguintes instruções de ida:

```
Siga norte a partir da Avenida NS da Penha 1600
Vire a direita na Avenida Maruipe
Vire a direita na Rua Guilherme Serrano
Vire a esquerda na Rua Vila Lobos
Chegou ao seu destino na Rua Vila Lobos 12
```

As instruções de retorno são as seguintes:

```
Siga oeste a partir da Rua Vila Lobos 12
Vire a direita na Rua Gilherme Serrano
Vire a esquerda na Avenida Maruipe
Vire a esquerda na Avenida NS da Penha
Chegou ao seu destino na Avenida NS da Penha 1600
```

As instruções devem seguir as regras da seguinte gramática:

<instrucoes> =	<start-instrucao> [<lista-instrucoes>] <end-instrucao>
<start-instrucao> =	Siga <direcao> a partir da <endereco>
<end-instrucao> =	Chegou ao seu destino na <endereco>
<endereco> =	<rua> <numero>
<lista-instrucoes> =	<instrucao> [<lista-instrucoes>]
<instrucao> =	Vire a <DE> na <rua>
<rua> =	<lista-palavras>
<lista-palavras> =	<palavra>
<lista-palavras> =	<palavra> [<lista-palavras>]
<DE> =	direita
<DE> =	esquerda
<direcao> =	norte
<direcao> =	sul
<direcao> =	leste
<direcao> =	oeste
Um <numero> é um inteiro de 1 a 999999 Uma <palavra> é uma string de letras e dígitos Um espaço simples separa palavras e números em uma linha	

Assuma que todas as ruas sejam mão-dupla e que tenham direção ou norte-sul ou leste-oeste.

A ENTRADA

A entrada consiste de um ou mais conjuntos de instruções. Cada conjunto de instruções começa com uma linha contendo um inteiro n (2<n<100 ou n = 0) que é o número total de instruções. Cada uma das n linhas seguintes contém uma instruções que deve seguir a gramática apresentada. Uma linha da instrução tem menos de 120 caracteres. O último conjunto de instruções tem n=0 e não deve ser processado.

A SAÍDA

A saída de cada conjunto de instruções deve começar com uma linha no formato:

```
Instrucoes <d>:
```

<d> é o número do conjunto de instruções, começando com 1. A seguir, n linhas de saída devem ser impressas, cada linha contendo uma instrução. Depois de cada conjunto de instruções deve haver uma linha em branco.

EXEMPLO DE ENTRADA

```
5
Siga norte a partir da Avenida NS da Penha 1600
Vire a direita na Avenida Maruipe
Vire a direita na Rua Guilherme Serrano
Vire a esquerda na Rua Vila Lobos
Chegou ao seu destino na Rua Vila Lobos 12
4
Siga norte a partir da Avenida Dante Micheline 26
Vire a esquerda na Rua Eugenilio Ramos
Vire a direita na Rua Ludwick Macal
Chegou ao seu destino na Rua Ludwick Macal 14
0
```

EXEMPLO DE SAÍDA

```
Instrucoes 1:
Siga oeste a partir da Rua Vila Lobos 12
Vire a direita na Rua Gilherme Serrano
Vire a esquerda na Avenida Maruipe
Vire a esquerda na Avenida NS da Penha
Chegou ao seu destino na Avenida NS da Penha 1600

Instrucoes 2:
Siga sul a partir da Rua Ludwick Macal 14
Vire a esquerda na Rua Eugenilio Ramos
```

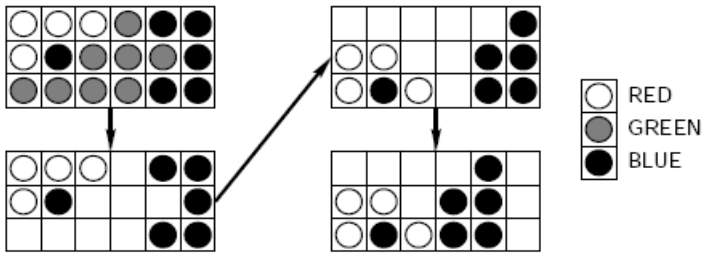
```
Vire a direita na Avenida Dante Micheline
Chegou ao seu destino na Avenida Dante Michenile 26
```

PROBLEMA D
O JOGO

O jogo de nome “Mesmo” é para se jogar sozinho num tabuleiro de 10 x 15. Cada quadrado contém uma bola vermelha (R), verde (G) ou azul (B). Duas bolas fazem parte do mesmo grupo se elas são da mesma cor e pode-se chegar a uma delas saindo da outra e seguindo bolas da mesma cor nas quatro direções esquerda, direita, pra cima e pra baixo. A cada etapa do jogo, o jogador escolhe uma bola cujo grupo contenha pelo menos duas bolas e retira essas bolas do grupo do tabuleiro. Então o tabuleiro é compressado em duas formas:

1. Desloque as bolas remanescentes de cada coluna para baixo para completar os espaços em branco. A ordem das bolas em cada coluna é preservada;
2. Se uma coluna ficar vazia, desloque as colunas remanescentes para a esquerda o máximo possível. A ordem das colunas é preservada.

Por exemplo, escolher a bola no canto inferior esquerdo do sub-tabuleiro abaixo causa:



O objetivo do jogo é remover todas as bolas do tabuleiro e o jogo termina quando todas as bolas são removidas ou quando resta apenas uma bola em cada grupo. A pontuação é como se segue. O jogador começa com 0 pontos. Quando um grupo de m bolas é removido, a pontuação do jogador cresce em (m-2)^2. Um bônus de 1000 é dado se cada bola é removida no final do jogo.

Você suspeita que uma boa estratégia possa ser escolher a bola que forneça o maior grupo possível a cada etapa, e você quer testar essa estratégia escrevendo um programa que simule jogos com essa estratégia. Se houver duas ou mais bolas para escolher, seu programa deve escolher a bola mais a esquerda que fornece o maior grupo. Se ainda houver empate, ele deve escolher a bola mais abaixo dentre as bolas mais a esquerda.

A ENTRADA

A primeira linha contém um número inteiro positivo que fornece o número de jogos a seguir. O arranjo inicial das bolas em cada jogo é dado linha por linha, de cima para baixo. Cada linha contém 15 caracteres, cada um sendo “R”, “G” ou “B”, especificando as cores das bolas na coluna da esquerda para a direita. Uma linha em branco precede cada jogo.

A SAÍDA

Para cada jogo, imprima o número do jogo, seguido de uma nova linha, depois a informação sobre cada movimento e finalmente pela pontuação final.

Move x at (r,c): removed b balls of color C, got s points.

Onde **x** é o número do movimento, **r** e **C** são os números da linha e da coluna da bola escolhida, respectivamente. As linhas são numeradas de 1 a 10 de baixo para cima e as colunas são numeradas de 1 a 15 a partir da esquerda. **b** é o número de bolas do grupo removido. **C** é uma entre “R”, “G” ou “B” indicando a cor das bolas removidas. **s** é a pontuação dessa jogada. A pontuação não inclui os 1000 pontos de bônus se todas as bolas são removidas depois da jogada.

A pontuação final deve ser gerada como se segue:

Final score: s, with b balls remaing.

Insira uma linha em branco entre as saídas de cada jogo. Use as formas plurais “balls” e “points” mesmo que o valor correspondente seja 1.

EXEMPLO DE ENTRADA

```
3

RGGBBGGRBRRGGBG
RBGRBGRBGRBGRBG
RRRRGBBBRGGRBBB
GGRGBGGBRRGGGBG
GBGGRRRRRBGGRRR
BBBBBBBBBBBBBBB
BBBBBBBBBBBBBBB
RRRRRRRRRRRRRRR
RRRRRRGGGGRRRRR
GGGGGGGGGGGGGGG

RRRRRRRRRRRRRRR
RRRRRRRRRRRRRRR
GGGGGGGGGGGGGGG
GGGGGGGGGGGGGGG
BBBBBBBBBBBBBBB
BBBBBBBBBBBBBBB
RRRRRRRRRRRRRRR
RRRRRRRRRRRRRRR
GGGGGGGGGGGGGGG
GGGGGGGGGGGGGGG
```

RBGRBGRBGRBGRBG
BGRBGRBGRBGRBGR
GRBGRBGRBGRBGRB
RBGRBGRBGRBGRBG
BGRBGRBGRBGRBGR
GRBGRBGRBGRBGRB
RBGRBGRBGRBGRBG
BGRBGRBGRBGRBGR
GRBGRBGRBGRBGRB
RBGRBGRBGRBGRBG

EXEMPLO DE SAÍDA

Game 1:

Move 1 at (4,1): removed 32 balls of color B, got 900 points.
Move 2 at (2,1): removed 39 balls of color R, got 1369 points.
Move 3 at (1,1): removed 37 balls of color G, got 1225 points.
Move 4 at (3,4): removed 11 balls of color B, got 81 points.
Move 5 at (1,1): removed 8 balls of color R, got 36 points.
Move 6 at (2,1): removed 6 balls of color G, got 16 points.
Move 7 at (1,6): removed 6 balls of color B, got 16 points.
Move 8 at (1,2): removed 5 balls of color R, got 9 points.
Move 9 at (1,2): removed 5 balls of color G, got 9 points.
Final score: 3661, with 1 balls remaining.

Game 2:

Move 1 at (1,1): removed 30 balls of color G, got 784 points.
Move 2 at (1,1): removed 30 balls of color R, got 784 points.
Move 3 at (1,1): removed 30 balls of color B, got 784 points.
Move 4 at (1,1): removed 30 balls of color G, got 784 points.
Move 5 at (1,1): removed 30 balls of color R, got 784 points.
Final score: 4920, with 0 balls remaining.

Game 3:

Final score: 0, with 150 balls remaining.

PROBLEMA E
DIGA QUEIJO!

Era uma vez, num pedaço gigante de queijo, uma aranhazinha chamada Amelia Aranha. Amelia deveria estar muito feliz, porque ela estava cercada pelo mais delicioso queijo que ela poderia comer. No entanto, ela sentia que algo faltava na sua vida.

Numa manhã, seus sonhos sobre queijo foram interrompidos por um barulho que nunca tinha ouvido antes. Mas ela imediatamente percebeu o que era – o som de uma aranha-macho mordendo seu pedaço de queijo!

Nada poderia parar Amelia agora. Ela tinha que conhecer aquela outra aranha o mais rápido possível. Para isso, ela deveria encontrar o caminho mais rápido para chegar à outra aranha. Amelia consegue comer um milímetro de queijo em dez segundos. Porém, o caminho em linha reta para a outra aranha pode ser o mais rápido. O queijo onde Amelia vive é cheio de buracos. Esses buracos, que são bolhas de ar aprisionadas no queijo, são esféricos na maioria dos casos. Mas, ocasionalmente, essas bolhas de ar sobrepõem-se, criando buracos compostos de todos os tipos. Amelia não gasta tempo para passar por um buraco no queijo, já que ela pode voar de um extremo ao outro instantaneamente. Logo, será útil viajar pelos buracos para chegar à outra aranha mais rapidamente.

Para esse problema, você deve escrever um programa que, dadas as posições das duas aranhas e dos buracos no queijo, determina o tempo mínimo gasto por Amelia pra chegar à outra aranha. Para isso, você pode assumir que o queijo é infinitamente largo. Isso porque o queijo é tal largo que nunca vale a pena para Amelia sair do queijo para alcançar a outra aranha. Você também pode assumir que a outra aranha está aguardando ansiosa a chegada de Amelia e não vai se mexer enquanto Amelia estiver no seu curso.

A ENTRADA

O arquivo de entrada contém a descrição de vários casos de teste. Cada caso começa com uma linha contendo um único inteiro n (0 <= n <= 100), o número de buracos no queijo. Essa é seguida por n linhas contendo quatro inteiros xi, yi, zi, ri cada. Esses descrevem os centros (xi, yi, zi) e os raios ri (ri >0) dos buracos. Todos os valores são dados em milímetros.

A descrição termina com duas linhas contendo três inteiros cada. A primeira contém os valores xa, ya, za, fornecendo a posição de Amelia no queijo, e a segunda contendo xo, yo, zo, fornecendo a posição da outra aranha.

O arquivo de entrada termina por uma linha contendo um -1.

A SAÍDA

Para cada caso de teste, imprima uma linha de saída, seguindo o formato do exemplo. Primeiro, imprima o número do caso de teste, começando por 1. Então imprima, o menor tempo necessário para Amelia alcançar a outra aranha, aproximando para o inteiro mais próximo.

A entrada será tal que a aproximação nunca será ambígua.

EXEMPLO DE ENTRADA

```
1
20 20 20 1
0 0 0
0 0 10
-1
```

EXEMPLO DE SAÍDA

```
Cheese 1: Travel time = 100 sec
```

PROBLEMA F
EVENTO CIENTÍFICO

Este problema foi elaborado pela Prof.ª Dr.ª Maria Claudia Silva Boeres (DI/UFES - ES) e pela Prof.ª Dr.ª Maria Cristina Rangel (DI/UFES – ES).

Para um evento científico estão programados cinco mini-cursos que devem ser distribuídos em cinco auditórios que possuem capacidades variadas. O número de alunos dos mini-cursos já está confirmado. A capacidade de cada auditório e o número de alunos de cada mini-curso estão na tabela abaixo:

Mini-cursos	Número de alunos	Auditórios	Capacidade
MC1	80	A1	50
MC2	30	A2	100
MC3	40	A3	40
MC4	75	A4	200
MC5	120	A5	150

Desenvolva um algoritmo que faça a alocação dos mini-cursos nos auditórios de forma que seja observada se existe solução para o problema, isto é, respeitando a capacidade de cada auditório e minimizando o desperdício de lugares vagos.

A ENTRADA

A primeira linha do arquivo de entrada é um inteiro com o número N_mc de mini-cursos do evento. Na próxima linha, lêem-se N_mc inteiros, que representam o número de alunos em cada mini-curso, na ordem dos mini-cursos. Como no exemplo, são 80 alunos para o mini-curso MC1, 30 para o MC2, e assim por diante.

Na próxima linha, o número de auditórios N_aud, onde (N_aud ≥ N_mc). Em seguida, as capacidades de cada auditório. Como para os mini-cursos, a capacidade dos auditórios também está em ordem, 50 para o A1, 100 para o A2 e assim por diante.

A SAÍDA

Para o arquivo de saída, considere a ordem numérica dos mini-cursos, ou seja, MC1, MC2, MC3, MC4 e MC5. Imprima em uma linha, para cada mini-curso MCi, o auditório Ai, com 1<= i <= 5, de acordo com a alocação gerada pelo seu programa.

Atenção, se não houver solução para o problema, imprima

Não há solução.

EXEMPLO DE ENTRADA

5 80 30 40 75 120 5 50 100 40 200 150
EXEMPLO DE SAÍDA
A5 A3 A1 A2 A4

PROBLEMA G
COMPRADOR DEPRESSIVO

Alguns meses atrás, Kara Van e Lee Sabre fizeram um empréstimo para a compra de um carro. Mas agora eles estão presos em casa num sábado à noite sem carro nem dinheiro. Houve um acidente, e a seguradora deu perda-total no carro. Por conta disso, eles receberam R$ 10.000,00, o valor corrente do carro. O único problema é que eles deviam R$ 15.000,00 ao banco, e esse exigiu o pagamento imediato, já que não havia mais um carro como garantia. Em apenas poucos instantes, esse casal não só perdeu um carro, como também R$ 5.000,00 também.

O que Kara e Lee esqueceram de considerar foi a depreciação, ou seja, a perda do valor do carro como o tempo. A cada mês, o pagamento do empréstimo reduz a dívida sobre o carro. Porém, a cada mês, o valor do carro também diminui. Sua tarefa é escrever um programa que calcula a primeira vez, medida em meses, que o comprador deve menos que o valor do carro. Para esse problema, a depreciação é especificada como uma porcentagem sobre o valor do carro no mês anterior.

A ENTRADA

A entrada consiste de informações sobre vários empréstimos. Cada empréstimo consiste de uma linha contendo a sua duração em meses, a mensalidade, o total do empréstimo e o número de porcentagens da depreciação. Todos os valores são positivos, a duração máxima dos empréstimos é de 100 meses e o valor máximo do carro é R$ 75.000,00. Já que a depreciação não é constante, as taxas variáveis são especificadas em uma série de registros de depreciação. Cada registro consiste em uma linha com o número do mês e a porcentagem de depreciação, maior que 0 e menor que 1. Eles estão em ordem crescente por mês, começando no mês 0. Essa depreciação é aplicada imediatamente depois de tirar o carro da concessionária e está sempre presente nos dados. Todas as outras porcentagens são correspondentes ao fim do mês. Nem todos os meses podem ser listados nos dados de entrada. Se um mês não é listado, então a depreciação anterior é aplicada a esse mês. O fim da entrada é assinalado com um tempo de duração de empréstimo negativo. Os outros três valores estarão presentes, porém indeterminados.

Para simplicidade, vamos assumir um empréstimo de 0% de juros, assim o valor inicial do carro é o valor total do empréstimo mais a mensalidade. É possível que os valores do carro e da dívida sejam positivos e menores que R$ 1,00. Não arredonde valores para um valor inteiro de centavos (R$ 7.652,365 não deve ser arredondado para 7.652,37).

Considere o exemplo abaixo de R$ 15.000,00 de empréstimo por 30 meses. Assim que o comprador sai da concessionária com o carro, ele ainda deve R$ 15.000,00 , mas o valor do carro caiu em 10% para R$ 13.950,00. Após quatro meses, o comprador efetuou quatro pagamentos, cada um de R$ 500,00 e o carro depreciou 3% nos meses 1 e 2 e 0.2% nos meses 3 e 4. Nesse momento, o carro vale R$ 13.073,10528 e o comprador deve apenas R$ 13.000,00.

A SAÍDA

Para cada empréstimo, é o número de meses completos antes que o comprador deva menos que o carro vale. Note que o inglês exige plural (5 months) em todos os valores diferentes de um (1 month).

EXEMPLO DE ENTRADA

```
30 500.0 15000.0 3
0 .10
1 .03
3 .002
12 500 9999.99 2
0 .05
2 .1
```

EXEMPLO DE SAÍDA

```
4 months
1 month
```

PROBLEMA H
A CRIPTOGRAFIA DE BOB

Este problema foi criado pelo Prof. Dr. João Paulo Andrade Almeirda, do Departamento de Informática da UFES.

Bob resolveu criar um procedimento para armazenar dígitos de 0 a 9 de forma criptografada.

O procedimento que ele criou é simples: O dobro do enésimo dígito de uma senha é somado ao enésimo dígito dos dados a serem criptografos; o último dígito do resultado desta soma é usado como enésimo dígito da mensagem criptografada. Como o tamanho da senha pode ser menor que o tamanho dos dados a serem criptografados uma senha mais longa é criada através da repetição da senha original até que a senha tenha o mesmo tamanho que os dados.

Faça um programa que criptografe e decriptografe dados de acordo com o procedimento de Bob. Uma entrada para o seu programa pode incluir vários comandos. A primeira linha de um comando indica qual a função a ser executada ("C" para criptografar, "D" para decriptografar, ou "FIM" para término do programa). A segunda linha indica o número de dígitos da senha a ser usada (no mínimo 1, no máximo 100). A terceira linha inclui os dígitos da senha (separados por espaço). A quarta linha indica o número de dígitos dos dados a serem criptografados ou decriptografados (no mínimo 1, no máximo 100). Finalmente, a quinta linha inclui o número de dígitos dos dados a serem criptografados ou decriptografados (separados por espaço).

A ENTRADA

No exemplo de entrada, o primeiro comando pede para que o programa criptografe os 11 primeiros dígitos de pi (31415926535) com a senha 8990 (que é expandida para 89908990899), resultando na seqüência 99211706113. Na continuação do exemplo no segundo comando, a seqüência criptografa é decriptografada resultando novamente nos 11 primeiros dígitos de pi (31415926535).

A SAÍDA

O seu programa deve imprimir produzir dados criptografados ou decriptografados de acordo com a entrada (uma linha por comando).

Atenção, pois a mensagem também pode ser menor que a senha.

EXEMPLO DE ENTRADA

```
C
4
```

```
8 9 9 0
11
3 1 4 1 5 9 2 6 5 3 5
D
4
8 9 9 0
11
9 9 2 1 1 7 0 6 1 1 3
C
5
1 2 3 4 5
10
0 1 2 3 4 5 6 7 8 9
FIM
```

EXEMPLO DE SAÍDA

```
9 9 2 1 1 7 0 6 1 1 3
3 1 4 1 5 9 2 6 5 3 5
2 5 8 1 4 7 0 3 6 9
1 4 7 0 3 6 9 2 5 8
```