CADERNO DE PROBLEMAS

19/05/2011 – das 14:00 às 18:00

Leia atentamente estas instruções antes de iniciar a resolução dos problemas. Este caderno é composto de 6 problemas, sendo que 2 deles estão descritos em inglês.

1. É permitido o uso de material impresso, livros, apostilas e dicionários bem como lápis, caneta ou lapiseira, borracha, régua e papel para rascunho. O acesso à internet é bloqueado durante a realização da prova. A ajuda do ambiente de desenvolvimento como C, C++ ou Java pode ser utilizada.
2. A correção das resoluções dos problemas será automatizada por meio do sistema de submissão eletrônica BOCA, 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 os nomes indicados nos problemas para nomear os seus arquivos-fonte, de acordo com a linguagem de programação utilizada.
6. Os problemas devem ser resolvidos utilizando o raciocínio entrada-processamento-saída, ou seja, não é necessário armazenar toda a saída para exibi-la.

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 C: `scanf`, `getchar`, `printf`, `putchar`;
- em C++: as mesmas de C ou os objetos `cout` e `cin`;
- em Java:

```
Scanner sc = new Scanner(); int i = sc.nextInt();
String s = sc.next(); float f = sc.nextFloat();
System.out.println(); System.out.printf("%d",i);
```

Lobo Mau (1 – verde)

Arquivo fonte: *lobo.c*, *lobo.cpp* ou *lobo.java*

Na fazenda do Sr. Amarante existe certo número de ovelhas. Enquanto elas estão dormindo profundamente, alguns lobos famintos tentam invadir a fazenda e atacar as ovelhas. Ovelhas normais ficariam indefesas diante de tal ameaça, mas felizmente as ovelhas do Sr. Amarante são praticantes de artes marciais e conseguem defender-se adequadamente.

A fazenda possui um formato retangular e consiste de campos arranjados em linhas e colunas. Cada campo pode conter uma ovelha (representada pela letra 'k'), um lobo (letra 'v'), uma cerca (símbolo '#') ou simplesmente estar vazio (símbolo '.'). Consideramos que dois campos pertencem a um mesmo pasto se podemos ir de um campo ao outro através de um caminho formado somente com movimentos horizontais ou verticais, sem passar por uma cerca. Na fazenda podem existir campos vazios que não pertencem a nenhum pasto. Um campo vazio não pertence a nenhum pasto se é possível "escapar" da fazenda a partir desse campo (ou seja, caso exista um caminho desse campo até a borda da fazenda).

Durante a noite, as ovelhas conseguem combater os lobos que estão no mesmo pasto, da seguinte forma: se em um determinado pasto houver mais ovelhas do que lobos, as ovelhas sobrevivem e matam todos os lobos naquele pasto. Caso contrário, as ovelhas daquele pasto são comidas pelos lobos, que sobrevivem. Note que caso um pasto possua o mesmo número de lobos e ovelhas, somente os lobos sobreviverão, já que lobos são predadores naturais, ao contrário de ovelhas.

Tarefa

Escreva um programa que, dado um mapa da fazenda do Sr. Amarante indicando a posição das cercas, ovelhas e lobos, determine quantas ovelhas e quantos lobos estarão vivos na manhã seguinte.

Entrada

A entrada contém vários conjuntos de testes, que devem ser lidos do dispositivo de entrada padrão (normalmente o teclado). A primeira linha da entrada contém dois inteiros R e C que indicam o número de linhas ($3 \leq R \leq 200$) e de colunas ($3 \leq C \leq 200$) de campos da fazenda. Cada uma das R linhas seguintes contém C caracteres, representando o conteúdo do campo localizado naquela linha e coluna (espaço vazio, cerca, ovelha ou lobo).

Exemplo de Entrada

```
6 6
...#..
.##v#.
#v.#.#
#.k#.#
.###.#
...###
8 8
.######.
#..k...#
#.####.#
#.#v.#.#
#.#.k#k#
#k.##..#
#.v..v.#
.######.
9 12
.###.#####..
#.kk#...#v#.
#..k#.#.#.#.
#..##k#...#.
#.#v#k###.#.
#..#v#....#.
#...v#v####.
.####.#vv.k#
.......#####
0 0
```

Saída

Seu programa deve imprimir, na saída padrão, uma única linha, contendo dois inteiros, sendo que o primeiro representa o número de ovelhas e o segundo representa o número de lobos que ainda estão vivos na manhã seguinte.

Exemplo de Saída

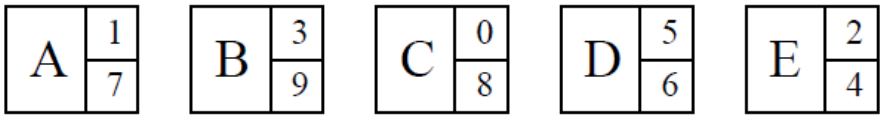
```
0 2
3 1
3 5
```

Proteja sua senha (2 – azul)

Source file: *senha.c*, *senha.cpp* ou *senha.java*

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 sequência 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 sequência de letras digitada, não é possível notar facilmente qual a senha do cliente do banco.

Dada uma sequência 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.

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 $N = 0$.

A entrada deve ser lida da entrada padrão.

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*", sendo *n* numerado sequencialmente 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 deve ser seguida rigorosamente.

A saída deve ser escrita na saída padrão.

Exemplo de entrada	**Saída para o 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	Teste 1 3 8 4 7 2 9 Teste 2 2 5 6 7 8 9

Temperatura Lunar (3 – laranja)

Arquivo fonte: *lua.c*, *lua.cpp* ou *lua.java*

Sem as proteções da atmosfera e do cinturão magnético que existem na Terra, a Lua fica exposta ao ataque do Sol, que é um astro em constante explosão atômica. As explosões do Sol emitem ondas letais de partículas. Uma pessoa que ficasse desprotegida na superfície da Lua, num lugar onde o Sol incidisse diretamente, sofreria um bombardeio radioativo tão intenso quanto se estivesse nas imediações da usina russa de Chernobyl no momento do acidente que matou 31 pessoas, em 1986. Além da radiação solar, outro efeito desta falta de proteção contra o Sol que existe na Lua é a enorme variação de temperatura. Nas regiões próximas do equador lunar, a variação de temperatura é brutal, passando de cerca de 130 graus positivos durante o dia a 129 graus negativos à noite.

Para estudar com mais precisão as variações de temperatura na superfície da Lua, a NASA enviou à Lua uma sonda com um sensor que mede a temperatura de 1 em 1 minuto. Um dado importante que os pesquisadores desejam descobrir é como se comporta a média da temperatura, considerada em intervalos de uma dada duração (uma hora, meia hora, oito horas, etc.). Por exemplo, para a sequência de medições 8, 20, 30, 50, 40, 20, -10, e intervalos de quatro minutos, as médias são respectivamente 108/4=27, 140/4=35, 140/4=35 e 100/4=25.

Você foi recentemente contratado pela NASA, e sua primeira tarefa é escrever um programa que, conhecidos a sequência de temperaturas medidas pelo sensor e o tamanho do intervalo desejado, informe qual a maior e qual a menor temperatura média observadas, considerando o tamanho do intervalo dado.

Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de teste contém dois números inteiros positivos N e M, que indicam respectivamente o número total de medições de temperatura de uma sequência obtida pelo sensor, e o tamanho dos intervalos, em minutos, em que as médias devem ser calculadas. As N linhas seguintes contêm um número inteiro cada, representando a sequência de medidas do sensor. O final da entrada é indicado quando $N = M = 0$.

A entrada deve ser lida da entrada padrão.

Saída

Para cada conjunto de teste da entrada seu programa deve produzir três linhas. A primeira linha identifica o conjunto de teste, no formato "Teste *n*", sendo *n* numerado a partir de 1. A segunda linha deve conter dois números inteiros, X e Y, separados por ao menos um espaço em branco, representando, respectivamente, os valores da menor e da maior média de temperatura, conforme determinado pelo seu programa. O valor da média deve ser truncado se a média não for um número inteiro (ou seja, deve ser impressa apenas a parte inteira). A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída deve ser seguida rigorosamente.

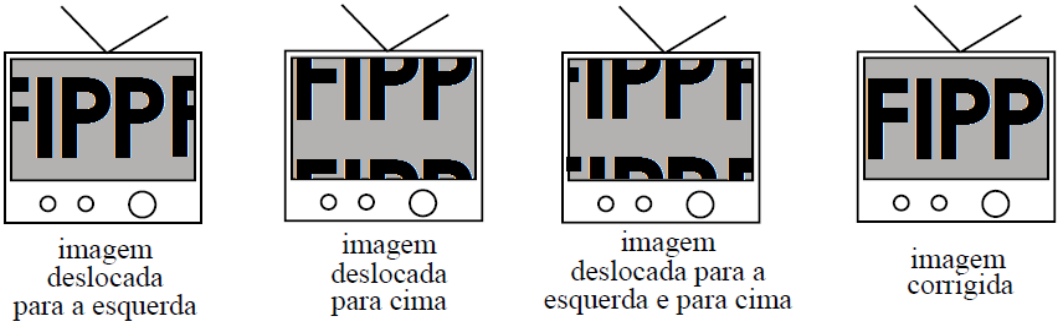
A saída deve ser escrita na saída padrão.

Exemplo de entrada	**Saída para o exemplo de entrada**
4 2 -5 -12 0 6 7 4 35 -35 5 100 100 50 50 0 0	Teste 1 -8 3 Teste 2 26 75

TV da Vovó (4 – vermelha)

Arquivo fonte: *tv.c*, *tv.cpp* ou *tv.java*

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 esquerda, a parte da imagem que deixa de ser vista à esquerda reaparece na tela do lado direito.



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 certo número de linhas (para cima ou para baixo) e certo número de colunas (para a direita ou para a esquerda).

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.

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 contêm cada uma N inteiros, descrevendo o valor de cada ponto da imagem. Após a descrição da imagem, segue 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 ($0 \leq X \leq 1000$) representa o deslocamento na direção horizontal (valor positivo representa deslocamento para a direita, valor negativo para a esquerda), e o valor de Y ($0 \leq Y \leq 1000$) 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 $M = N = 0$.

A entrada deve ser lida da entrada padrão.

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*", sendo *n* numerado sequencialmente 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.

A saída deve ser escrita na saída padrão.

Exemplo de entrada	**Saída para o 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`
	`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`

Parking (5 – branca)

Source file name: *parking.c*, *parking.cpp* ou *parking.java*

Description

When shopping on Long Street, Michael usually parks his car at some random location, and then walks to the stores he needs. Can you help Michael choose a place to park which minimizes the distance he needs to walk on his shopping round?

Long Street is a straight line, where all positions are integer. You pay for parking in a specific slot, which is an integer position on Long Street. Michael does not want to pay for more than one parking though. He is very strong, and does not mind carrying all the bags around.

Input

The first line of input gives the number of test cases, $1 \leq t \leq 100$. There are two lines for each test case. The first gives the number of stores Michael wants to visit, $1 \leq n \leq 20$, and the second gives their n integer positions on Long Street, $0 \leq x_i \leq 99$.

The input must be read from standard input.

Output

Output for each test case a line with the minimal distance Michael must walk given optimal parking.

The output must be written to standard output.

Sample input	Output for the sample input
2 4 24 13 89 37 6 7 30 41 14 39 42	152 70

The 3n + 1 problem (6 – amarela)

Arquivo fonte: *problem.c*, *problem.cpp* ou *problem.java*

Problems in Computer Science are often classified as belonging to a certain class of problems (e.g., NP, Unsolvable, Recursive). In this problem you will be analyzing a property of an algorithm whose classification is not known for all possible inputs.

The Problem

Consider the following algorithm:

1. input n
2. print n
3. if $n = 1$ then STOP
4. if n is odd then $n \leftarrow 3x + 1$
5. else $n \leftarrow n/2$
6. GOTO 2

Given the input 22, the following sequence of numbers will be printed 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

It is conjectured that the algorithm above will terminate (when a 1 is printed) for any integral input value. Despite the simplicity of the algorithm, it is unknown whether this conjecture is true. It has been verified, however, for all integers n such that $0 < n <$ 1,000,000 (and, in fact, for many more numbers than this.)

Given an input n, it is possible to determine the number of numbers printed (including the 1). For a given n this is called the cycle-length of n. In the example above, the cycle length of 22 is 16.

For any two numbers i and j you must determine the maximum cycle length over all numbers between i and j.

Input

The input will consist of a series of pairs of integers i and j, one pair of integers per line. All integers will be less than 1,000,000 and greater than 0.

You must process all pairs of integers and for each pair determine the maximum cycle length over all integers between and including i and j.

You can assume that no operation overflows a 32-bit integer.

The input must be read from standard input.

Output

For each pair of input integers i and j you should output i, j, and the maximum cycle length for integers between and including i and j. These three numbers should be separated by at least one space with all three numbers on one line and with one line of

output for each line of input. The integers *i* and *j* must appear in the output in the same order in which they appeared in the input and should be followed by the maximum cycle length (on the same line).

The output must be written to standard output.

Sample input	**Output for the sample input**
1 10 100 200 201 210 900 1000 0 0	1 10 20 100 200 125 201 210 89 900 1000 174