CADERNO DE PROBLEMAS

22/05/2015 – 14h às 17h30

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. Não é permitido o uso de material impresso, livros, apostilas e dicionários. Apenas é permitido o uso de lápis, caneta, lapiseira, borracha, régua e papel para rascunho (o papel é fornecido pela comissão organizadora). 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 `cin` e `cout`;
- 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);
```

Em C ou C++: use `sprintf(str,"%d",num);` em vez de `itoa(num,str,10);`

Em Java: não utilize `package`.

Faculdade de Informática de Presidente Prudente

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

Amigos ou Inimigos? (vermelha)

Problema
A

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

Um determinado exército numa certa fronteira decidiu enumerar as coordenadas em sua volta, de maneira a tornar difícil para o inimigo saber a quais posições eles estão se referindo, no caso de o sinal de rádio usado para comunicação ser interceptado. O processo de enumeração escolhido foi o seguinte: primeiro decide-se onde ficam os eixos x e y; a seguir, define-se uma equação linear que descreva a posição da fronteira em relação aos eixos (sim, ela é uma linha reta); finalmente, enumeram-se todos os pontos do plano cartesiano que não fazem parte da fronteira, sendo o número 0 atribuído à coordenada (0,0) e daí em diante atribuindo-se os números para as coordenadas inteiras seguindo uma espiral de sentido horário, sempre pulando os pontos que caem em cima da fronteira (veja a Figura 1). Caso o ponto (0,0) caia em cima da fronteira, o número 0 é atribuído ao primeiro ponto que não faça parte da fronteira seguindo a ordem especificada.

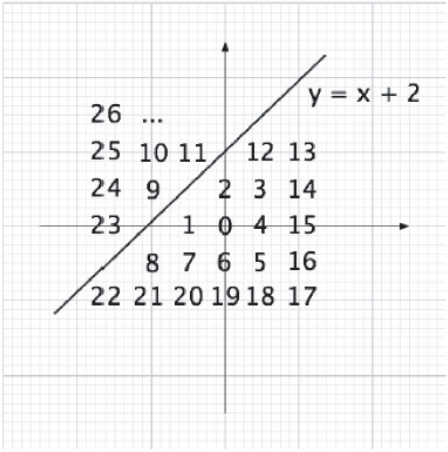


Figura 1. Enumeração dos pontos das coordenadas inteiras.

De fato, o inimigo não tem como saber a qual posição o exército se refere, a não ser que o inimigo saiba o sistema usado para enumerar os pontos. Tal esquema, porém, complicou a vida do exército, uma vez que é difícil determinar se dois pontos quaisquer estão no mesmo lado da fronteira ou em lados opostos. É aí que eles precisam da sua ajuda.

Entrada

A entrada contém vários casos de teste. A primeira linha da entrada contém um inteiro N ($1 \leq N \leq 100$) que representa a quantidade de casos de teste. Seguem-se os N casos de teste. A primeira linha de cada caso de teste contém dois inteiros a e b ($-5 \leq a \leq 5$ e $-10 \leq b \leq 10$), que descrevem a equação da fronteira: $y = ax + b$. A segunda linha de cada caso de teste contém um inteiro K, indicando a quantidade de consultas que se seguem ($1 \leq K \leq 1000$). Cada uma das K linhas seguintes descreve uma consulta, sendo composta por dois inteiros M e N representando as coordenadas enumeradas de dois pontos ($0 \leq M, N \leq 65535$).

A entrada deve ser lida da entrada padrão.

Saída

Para cada caso de teste da entrada seu programa deve produzir $K + 1$ linhas. A primeira linha deve conter a identificação do caso de teste na forma `Caso X`, sendo que `X` deve ser substituído pelo número do caso (iniciando de 1). As K seguintes devem conter os resultados das K consultas feitas no caso correspondente da entrada, na forma:

```
Mesmo lado da fronteira
```

ou

```
Lados opostos da fronteira
```

A saída deve ser escrita na saída padrão.

Exemplo de entrada	**Saída para o exemplo de entrada**
2 1 2 10 26 25 25 11 24 9 23 28 25 9 25 1 25 0 9 1 23 12 26 17 1 2 12 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 10 11 11 12	Caso 1 Mesmo lado da fronteira Mesmo lado da fronteira Mesmo lado da fronteira Mesmo lado da fronteira Mesmo lado da fronteira Lados opostos da fronteira Lados opostos da fronteira Lados opostos da fronteira Lados opostos da fronteira Lados opostos da fronteira Caso 2 Mesmo lado da fronteira Mesmo lado da fronteira Mesmo lado da fronteira Mesmo lado da fronteira Mesmo lado da fronteira Mesmo lado da fronteira Mesmo lado da fronteira Mesmo lado da fronteira Mesmo lado da fronteira Lados opostos da fronteira Mesmo lado da fronteira Mesmo lado da fronteira Lados opostos da fronteira

Circuito Bioquímico Digital (amarela)

Problema B

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

Um circuito bioquímico digital (CBD) é um artefato composto de um conjunto de *pontos de processamento*. Cada ponto de processamento é constituído por um minúsculo receptáculo para reagentes bioquímicos, feito de um substrato biológico que se comporta como um microcircuito eletrônico digital. Dependendo do estado da reação no receptáculo, o substrato gera dois níveis de voltagem. Um leitor acoplado ao CBD é capaz de realizar a leitura de todos os pontos de processamento de um CDB num dado instante, interpretando os dois níveis de voltagem como 0 ou 1.

Um experimento com o CBD é realizado da seguinte maneira. Os pontos de processamento são carregados com as substâncias de interesse e reagentes apropriados e, a cada intervalo fixo de tempo (tipicamente alguns milissegundos), os pontos de processamento são lidos. Assim, o experimento resulta em uma sequência de conjuntos (vetores) de bits, cada vetor correspondendo a uma medição do CBD.

Uma sequência ininterrupta de bits 1 de um mesmo ponto de processamento ao longo do tempo é denominada de palito. O comprimento de um palito é o número de bits 1 que o compõe (note que o comprimento dos palitos de um experimento pode variar entre um e o número de medições efetuadas). Uma característica importante de um experimento com o CBD é a quantidade e o comprimento dos palitos gerados. A figura abaixo mostra o resultado de um experimento realizado com um CBD de seis pontos de processamento, em que foram efetuadas quatro medições, contendo três palitos de comprimento um, um palito de comprimento dois e um palito de comprimento quatro.

```
0 1 0 1 1 0
0 0 0 1 0 0
0 1 0 1 0 1
0 1 0 1 0 0
```

Você foi contratado para escrever um programa que determine, dado o resultado de um experimento, quantos palitos com comprimento igual ou maior a certo valor foram gerados.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém três inteiros P, N e C, que indicam respectivamente o número de pontos de processamento ($1 \leq P \leq 1000$), o número de medições efetuadas ($1 \leq N \leq 1000$) e o comprimento mínimo de palitos de interesse ($1 \leq C \leq N$). Cada uma das próximas N linhas contém sequências de P dígitos $\{0, 1\}$, separados por um espaço em branco. O final da entrada é indicado por $P = N = C = 0$.

A entrada deve ser lida da entrada padrão.

Saída

Para cada caso de teste da entrada seu programa deve produzir uma única linha da saída, contendo o número de palitos de comprimento maior ou igual a *C* produzidos pelo experimento.

A saída deve ser escrita na saída padrão.

Exemplo de entrada	**Saída para o exemplo de entrada**
2 2 2 1 1 1 1 4 5 3 0 1 0 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 0 0 0 0 0	2 2

Polynomial Printer (azul)

Problema C

Source file: *polynomial.c*, *polynomial.cpp* or *polynomial.java*

It is given a sequence of integers (a_n, …, a_0), which represents polynomial having these coefficients:

$$a_n x^n + a_{n-1} x^{n-1} + ... + a_0$$

For example, the sequence (3, 4, 2, 5) represents the polynomial

$$3x^3 + 4x^2 + 2x + 5$$

The program prints the polynomial in a nice looking format. Allowing the carat character ('^', which is located above the '6' key on most keyboards) to represent exponentiation, the above polynomial can be expressed as follows:

$$3x\text{^}3 + 4x\text{^}2 + 2x + 5$$

This sounds easy, but there are dangers! For example, for the sequence (4, –1, 0) do you really want to print the following?

$$4x\text{^}2 + -1x + 0$$

No. Instead, you want to print:

$$4x\text{^}2 - x$$

There are other dangers too but we won't tell you what they are. Note that format is important, and should match the above form exactly (including where spaces appear). More examples are given below.

Samples

The input will consist of a sequence of lines. Each line contains the coefficients for one polynomial. The coefficients are integers and are separated by spaces. Note that leading 0's are possible (as seen in the last example below).

Input:	Polynomial:
3 4 2 5	3x^3 + 4x^2 + 2x + 5
4 -1 0	4x^2 - x
-1 1 -1	-x^2 + x - 1
-4 -2 003 -5	-4x^3 - 2x^2 + 3x - 5
-8	-8
0	0
0 -3 0	-3x

Input

The input contains several test cases. The first row entry contains an integer N ($1 \leq N \leq 100$) representing the amount of test cases. Each line of the input contains a sequence of integer coefficients, separated by spaces. The first coefficient belongs to the highest order coefficient and the last belongs to the lowest (0^{th} order) coefficient.

The input must be read from standard input.

Output

For each input line the output consists of the original input line and the final polynomial.
The output must be written to standard output.

Sample Input	**Output for the sample Input**

`2`
`3 4 2 5`
`4 -1 0`
	`Input: 3 4 2 5`
	`Polynomial: 3x^3 + 4x^2 + 2x + 5`
	`Input: 4 -1 0`
	`Polynomial: 4x^2 - x`

Secret Message Pad (verde)

Problema D

Source file: *secret.c*, *secret.cpp* or *secret.java*

Hearing stories from the Oracle that a programmer named Neo may possibly be "The One", Trinity risks going into to Matrix to observe Neo. While in the Matrix, her communications are intercepted and Trinity is almost caught by Agent Smith and two other agents.

Safely back in the hovercraft Nebuchadnezzar, Trinity decides to add another level of encryption to her communication using a "on-time pad". The idea is that given a sequence of numbers (a pad), any message can be proven to be securely encrypted if each character in the string is encoded by a number from the pad, if each number from the pad is only used once.

Trinity decides to use the one-time pad to encrypt her messages by shifting each letter in the message by k positions later in the alphabet, where k is determined by the one-time pad. Shifts occur in alphabetical order, between the letters "a" and "z". If a letter is shifted past "z", it starts back at "a" and continues shifting. For example, shifting each letter by $k = 2$, the word "car" is transformed into "ect", while the word "yaz" is shifted to "acb".

Input

The input will begin with the size of the one-time pad, followed by a sequence of numbers for the pad. The remaining input consists of a series of words to be encrypted using the keyword. The input will be terminated by a line containing only –1.

You may assume that the maximum size of the pad is 100 numbers, all numbers in the pad are between 0 and 25, and that all input will be lowercase letters.

The input must be read from standard input.

Output

For each word to be encrypted, output a line containing the encrypted word.

The output must be written to standard output.

Sample input	**Output for the sample input**
2 10 1 2 3 4 5 4 3 2 1 0 aaaaa zzzzz -1 40 1 5 2 21 3 8 4 25 11 9 6 7 8 9 11 12 3 0 11 4 14 21 9 0 1 3 12 7 2 11 5 9 20 12 1 19 4 9 8 24 humans make good batteries gnlr esrf -1	bcdef dcbaz izovqa qzvn mvwm mmwtpvwzb good luck

Loteria de Final de Semana (rosa)

Problema E

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

Algumas pessoas são contra loterias por motivos morais, alguns governos proibiram as loterias, mas com o advento da Internet esta forma popular de jogos de azar, que começou na China e ajudou a financiar a Grande Muralha, está prosperando.

Mas as chances de ganhar na loteria nacional são pequenas e, portanto, seus colegas de faculdade decidiram organizar uma loteria privada, com sorteios toda sexta-feira. A loteria é baseada em um estilo popular: um estudante que quer apostar escolhe C números distintos de 1 a K e paga R$ 1,00 (note que as loterias tradicionais como a Mega-Sena usam $C = 6$ e $K = 60$). Na sexta-feira durante o almoço C números (também de 1 a K) são sorteados. O estudante cuja aposta tem o maior número de acerto recebe o valor arrecadado nas apostas. Este montante é dividido no caso de empates e acumulado para a próxima semana se ninguém adivinhou qualquer um dos números sorteados. Alguns de seus colegas não acreditam nas leis da probabilidade e pediu-lhe para escrever um programa que determina os números que foram sorteados o menor número de vezes, considerando todos os sorteios anteriores, para que eles possam apostar nesses números.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém três inteiros N, C e K, que indicam, respectivamente, o número de sorteios que já tenham ocorrido ($1 \leq N \leq 10000$), quantos números compreendem uma aposta ($1 \leq C \leq 10$) e o máximo valor dos números a serem escolhidos na aposta ($C < K \leq 100$). Cada uma das próximas N linhas contém C inteiros distintos Xi indicando os números sorteados em cada concurso anterior ($1 \leq Xi \leq K$, para $1 \leq i \leq C$). O final da entrada é indicado por $N = C = K = 0$. *A entrada deve ser lida da entrada padrão.*

Saída

Para cada caso de teste da entrada seu programa deve escrever uma linha de saída, contendo o conjunto de números que foram sorteados o menor número de vezes. Este conjunto deve ser impresso como uma lista, por ordem crescente de números. Deixe um espaço em branco entre dois números consecutivos na lista. *A saída deve ser escrita na saída padrão.*

Exemplo de entrada	**Saída para o exemplo de entrada**
5 4 6 6 2 3 4 3 4 6 5 2 3 6 5 4 5 2 6 2 3 6 4 4 3 4 3 2 1 2 1 4 4 3 2 1 4 3 0 0 0	1 1 2 3 4

Zona de Animais (branca)

Problema F

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

Você foi convidado a conceber um novo zoológico, onde cada espécie animal deve viver em sua própria zona de animais retangular. Você convocou diferentes empresas de design para projetar a configuração de zonas dos animais. Todas as empresas irão apresentar as coordenadas para as zonas dos animais, e a construção do novo zoológico começará quando você tiver a configuração das zonas sem se sobrepor.

Você foi contratado para escrever um programa que analisa os limites das zonas e relata se quaisquer zonas de animais se sobrepõem. Duas zonas são consideradas sobrepostas se uma zona contém parte da outra, se elas compartilham uma porção de uma lateral em comum, ou se elas compartilham um canto em comum.

Cada empresa vai apresentar as coordenadas (x, y) do canto inferior esquerdo e superior direito de cada zona de animal, onde (0, 0) indica o canto inferior esquerdo do zoológico. Você pode assumir que todas as zonas devem caber dentro de uma área de 20 por 20 (coordenadas variam entre 0..19).

Entrada

A entrada é composta de uma linha contendo o número N de zonas de animais, seguido por uma linha para cada zona no formato: x_1 y_1 x_2 y_2 onde (x_1, y_1) é o canto inferior esquerdo do retângulo e (x_2, y_2) é o canto superior direito do retângulo. O final da entrada é indicado por $N = 0$.

A entrada deve ser lida da entrada padrão.

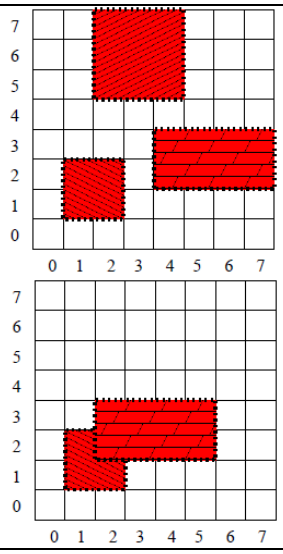
Saída

A saída será "Overlap" se as zonas se sobrepõem, caso contrário, "No Overlap".

A saída deve ser escrita na saída padrão.

Exemplo de entrada

```
3
1 1 2 2
2 5 4 7
4 2 7 3
2
1 1 2 2
2 2 5 3
0
```



Saída para o exemplo de entrada

```
No Overlap
Overlap
```