8º Torneio de Programação de Computadores

Contest

Organização:

Patrocínio:

Apoio:

Data: 29 de maio de 2010
Local: Laboratório de Informática - CT IX
Duração: 14:00 às 19:00

Divisibility

Nome do arquivo fonte: divisibility.{c,cpp,java}

Consider an arbitrary sequence of integers. One can place + or - operators between integers in the sequence, thus deriving different arithmetical expressions that evaluate to different values. Let us, for example, take the sequence: 17, 5, -21, 15. There are eight possible expressions:

17 + 5 + -21 + 15 = 16

17 + 5 + -21 - 15 = -14

17 + 5 - -21 + 15 = 58

17 + 5 - -21 - 15 = 28

17 - 5 + -21 + 15 = 6

17 - 5 + -21 - 15 = -24

17 - 5 - -21 + 15 = 48

17 - 5 - -21 - 15 = 18

We call the sequence of integers divisible by K if + or - operators can be placed between integers in the sequence in such way that resulting value is divisible by K. In the above example, the sequence is divisible by 7 (17+5+-21-15=-14) but is not divisible by 5.

You are to write a program that will determine divisibility of sequence of integers.

Input:

The first line of the input file contains a integer M indicating the number of cases to be analyzed. Then M couples of lines follow. For each one of this couples, the first line contains two integers, N and K (1 <= N <= 10000, 2 <= K <= 100) separated by a space. The second line contains a sequence of N integers separated by spaces. Each integer is not greater than 10000 by it's absolute value.

Ouput:

For each case in the input file, write to the output file the word "Divisible" if given sequence of integers is divisible by K or "Not divisible" if it's not.

Samples:

Input:	**Output:**
2 4 7 17 5 -21 15 4 5 17 5 -21 15	Divisible Not divisible

Divisibilidade

Nome do arquivo fonte: divisibility.{c,cpp,java}

Considere uma seqüência arbitrária de números inteiros. Pode-se colocar os operadores de + ou - entre inteiros na seqüência, obtendo, assim diferentes expressões aritméticas que avaliam a valores diferentes. Vejamos, por exemplo, a seqüência: 17, 5, -21, 15. Há oito possíveis expressões:

17 + 5 + -21 + 15 = 16

17 + 5 + -21 - 15 = -14

17 + 5 - -21 + 15 = 58

17 + 5 - -21 - 15 = 28

17 - 5 + -21 + 15 = 6

17 - 5 + -21 - 15 = -24

17 - 5 - -21 + 15 = 48

17 - 5 - -21 - 15 = 18

Chamamos a seqüência de números inteiros divisíveis por K se os operadores + ou - podem ser colocados entre inteiros na seqüência de tal forma que o valor resultante é divisível por K. No exemplo acima, a seqüência é divisível por 7 (17+5+-21-15=-14), mas não é divisível por 5.

Você deve escrever um programa que irá determinar a seqüência de divisibilidade de números inteiros.

Entrada:

A primeira linha do arquivo de entrada contém um inteiro M, indicando o número de casos a serem analisados.Depois M duplas de linhas abaixo. Para cada uma dess duplas, a primeira linha contém dois inteiros, N e K (1 <= N <= 10000, 2 <= K <= 100), separadas por um espaço.A segunda linha contém uma seqüência de N inteiros separados por espaços.Cada inteiro não é maior do que 10000 pelo seu valor absoluto.

Saída:

Para cada caso no o arquivo de entrada, escreva para o arquivo de saída a palavra "Divisible" se a seqüência de números inteiros é dada é divisível por K ou "Not divisible" se não é.

Exemplos

Entrada:	**Saída:**
2 4 7 17 5 -21 15 4 5 17 5 -21 15	Divisible Not divisible

Urn-ball probabilities!

Nome do arquivo fonte: urnball.{c,cpp,java}

Assume that you have two urns before you. Initially, one urn has one ball and the other urn has two balls and exactly one ball in each urn is red. At this initial stage you are asked to pick up two balls, one from each urn. Then one white ball is added in each urn and you are again asked to pick up one ball from each urn then again one white ball is added in each urn. This process continues for a certain time. Remember that you place the picked ball back to the urn after each pick up. You will have to determine the probability that in any of your pickups both of the picked balls were red and also the probability that all of your picked balls were red after certain steps.

Input:

The input file contains several lines of inputs. Each line of the input file contains an unsigned integer N (N<1000000) indicating how many times you will pick up. Of course after each pick up an increment in balls occurs as described previously.

Output:

For each line of input print a single line of output containing a floating point number and an integer. The floating-point number indicates the probability that you have picked up two red balls in at least one of your pick-ups and the second integer denotes how many consecutive zeros are there after decimal point in the probability value that all of your pick ups has both balls as red.

Samples:

Input:	**Output:**
1 2 20	0.500000 0 0.583333 1 0.688850 38

Urn-ball probabilidades!

Nome do arquivo fonte: urnball.{c,cpp,java}

Suponha que exista duas urnas à sua frente. Inicialmente, uma urna tem uma bola e a outra tem duas bolas. Exatamente uma bola de cada urna é vermelha. Nesta fase inicial você está convidado a pegar duas bolas, uma de cada urna. Então uma bola branca é adicionado em cada urna e você está novamente convidado a pegar uma bola de cada urna, em seguida, novamente uma bola branca é adicionado em cada urna. Esse processo continua por um tempo determinado. Lembre-se de colocar a bola de volta na urna antes de pegar outra. Você terá que determinar, após um certo número de passos, a probabilidade de as duas bolas retiradas serem vermelhas, e também a probabilidade de todas as bolas retiradas serem vermelhas em todas as suas retiradas.

Entrada:

O arquivo de entrada contém várias linhas de entradas.Cada linha do arquivo de entrada contém um inteiro sem sinal N (N <1000000) indica quantas vezes você vai retirar as bolas das urnas. É claro que depois de cada retirada um incremento nas bolas ocorre, como descrito anteriormente.

Saída:

Para cada linha da entrada uma única linha de saída que contém um número de ponto flutuante e um número inteiro.O número de ponto flutuante que indica a probabilidade de você ter pego duas bolas vermelhas em pelo menos uma de suas retiradas e o segundo inteiro indica quantos zeros existem consecutivos após o ponto decimal no valor de probabilidade de que todas suas retiradas terem sido ambas as bolas vermelhas.

Exemplos:

Entrada:	**Saída:**
1 2 20	0.500000 0 0.583333 1 0.688850 38

Connect the Campus

Nome do arquivo fonte: connect.{c,cpp,java}

Many new buildings are under construction on the campus of the University of Waterloo. The university has hired bricklayers, electricians, plumbers, and a computer programmer. A computer programmer? Yes, you have been hired to ensure that each building is connected to every other building (directly or indirectly) through the campus network of communication cables.

We will treat each building as a point specified by an x-coordinate and a y-coordinate. Each communication cable connects exactly two buildings, following a straight line between the buildings. Information travels along a cable in both directions. Cables can freely cross each other, but they are only connected together at their endpoints (at buildings).

You have been given a campus map which shows the locations of all buildings and existing communication cables. You must not alter the existing cables. Determine where to install new communication cables so that all buildings are connected. Of course, the university wants you to minimize the amount of new cable that you use.

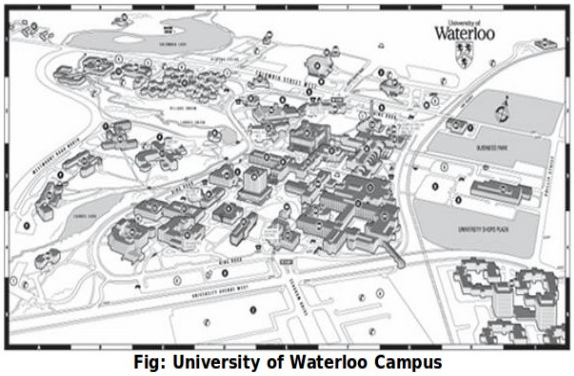


Fig: University of Waterloo Campus

Input:

The input file describes several test case. The description of each test case is given below:

The first line of each test case contains the number of buildings N (1<=N<=750). The buildings are labeled from 1 to N. The next N lines give the x and y coordinates of the buildings. These coordinates are integers with absolute values at most 10000. No two buildings occupy the same point. After that there is a line containing the number of existing cables M (0 <= M <= 1000) followed by M lines describing the existing cables. Each cable is represented by two integers: the building numbers which are directly connected by the cable. There is at most one cable directly connecting each pair of buildings.

Output:

For each set of input, output in a single line the total length of the new cables that you plan to use, rounded to two decimal places.

Samples:

Input:	Output:
4 103 104 104 100 104 103 100 100 1 4 2 4 103 104 104 100 104 103 100 100 1 4 2	4.41 4.41

Ligue o Campus

Nome do arquivo fonte: connect.{c,cpp,java}

Muitos edifícios novos estão em construção no campus da Universidade de Waterloo.A universidade contratou pedreiros, eletricistas, encanadores, e um programador de computador. Um programador de computador? Sim, você foi contratado para garantir que cada edifício esteja ligado a cada outro edifício (direta ou indiretamente), através da rede de cabos de comunicação do campus.

Vamos tratar cada edifício como um ponto especificado por uma coordenada x e uma coordenada y. Cada cabo de comunicação conecta exatamente dois edifícios, seguindo uma linha reta entre os prédios. A informação viaja ao longo de um cabo em ambas as direções. Os cabos podem livremente se cruzarem, mas são ligados entre si apenas em suas extremidades (em edifícios).

Você tem um mapa do campus, que mostra as localizações de todos os edifícios e os cabos de comunicação existentes.Você não deve alterar os cabos existentes. Determine onde instalar cabos de comunicação para que todos os novos edifícios estão ligados. Naturalmente, a universidade quer minimizar a quantidade de novos cabos que você usa.

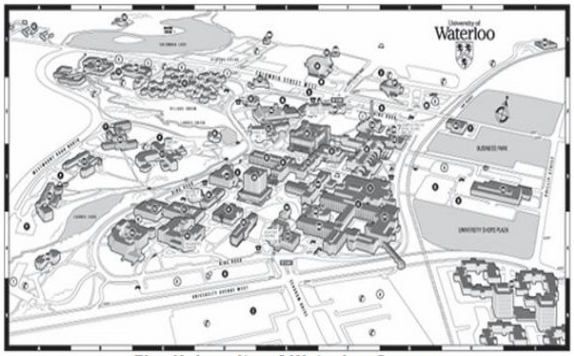


Fig: University of Waterloo Campus

Entrada:

O arquivo de entrada descreve vários casos de teste.A descrição de cada caso de teste é dada abaixo:

A primeira linha de cada caso de teste contém o número de edifícios N (1 <= N <= 750).Os edifícios são rotulados de 1 a N. As próximas linhas N fornecem as coordenadas X e Y dos edifícios. Essas coordenadas são números inteiros com valores absolutos, no máximo, 10000. Não existem dois edifícios que ocupam o mesmo ponto. Depois existe uma linha contendo o número de cabos existentes M (0 <= M <= 1000) seguido de M linhas descrevendo os cabos existentes. Cada cabo é representado por dois números inteiros: o número de construções que estão diretamente ligados pelo cabo. Há, no máximo, um cabo que conecta diretamente cada par de edifícios.

Saída:

Para cada conjunto de entrada a saída será uma única linha, o comprimento total dos novos cabos que você pretende usar, arredondado para duas casas decimais.

Exemplos:

Entrada:	**Saída:**
4 103 104 104 100 104 103 100 100 1 4 2 4 103 104 104 100 104 103 100 100 1 4 2	4.41 4.41

Viajando pelo espaço sideral

Nome do arquivo fonte: sideral.{c,cpp,java}

Para que se movam as gigantescas naves espaciais, é preciso de muita energia. Essa energia é coletada da luz das estrelas porque todos sabem como é perigoso o uso de energia nuclear. A intensidade da luz de uma estrela determina até que distância máxima ela consegue dar suporte a uma nave, de forma que uma estrela de fraca intensidade só consegue gerar energia suficiente para uma nave bem próxima enquanto estrelas distantes dão suporte a naves a uma distância impressionante. Uma implicação direta disso é que há algumas áreas escuras do espaço por onde naves não podem passar. Sua tarefa é informar se é possível que uma nave se mova de uma posição inicial até uma posição final no espaço sideral, sempre sendo abastecida por energia proveniente de uma estrela (sem passar por áreas escuras).

Há uma importante restrição que facilita seu trabalho. As naves não podem circular livremente pelo espaço tridimensional. Para evitar colisões em pleno espaço, uma Convenção Intergaláctica determinou diversos planos bidimensionais por onde as naves podem circulam. Como ainda não se sabe se o universo é finito ou está em constante crescimento, cada um desses planos é limitado por coordenadas que variam de uma origem simbólica no ponto de coordenadas (0,0) até o ponto mais distante de coordenadas (10.000, 10.000). Em outros termos: qualquer nave está sempre em um ponto (X,Y) de um plano cartesiano de forma que 0<=X<=10.000 e 0<=Y<=10.000. Como se sabe, as estrelas possuem um centro de energia, que também está localizado nesse plano bidimensional, ou seja, o centro de todas as estrelas também está limitado por coordenadas variando de 0 a 10.000 (inclusive).

Entrada :

A entrada é formada por vários casos de teste. Cada caso de teste representa um plano bidimensional, com a localização das várias estrelas naquele plano, a posição inicial da nave e qual a sua posição de destino desejada. Cada caso inicia por um inteiro E (1<=E<=100) informando quantas estrelas há naquele plano bidimensional. Quando E for informado igual a 0 (zero), o programa deve terminar imediatamente sem processar aquela entrada. A seguir, aparecem E linhas na entrada, cada uma delas com três inteiros X, Y e R (0<=X<=10.000, 0<=Y<=10.000 e 0<=R<=5.000) indicando o centro de energia da estrela (X,Y) e o raio máximo (R) a partir daquele centro que a estrela consegue abastecer uma nave. Depois disso, mais duas linhas na entrada apresentam respectivamente a posição inicial da nave como dois inteiros (Xi, Yi) e a posição de destino almejada para a nave como dois inteiros (Xd, Yd). Os inteiros Xi, Yi, Xd e Yd variam de 0 a 10.000, inclusive. A entrada deve ser lida da entrada padrão.

Saída :

Para cada plano bidimensional lido na entrada, o programa deve exibir uma única linha na saída contendo o texto “sim” quando for possível viajar do ponto (Xi, Yi) até o ponto (Xd, Yd) em qualquer percurso que esteja abastecido pela luz de estrelas (e não necessariamente uma linha reta!). Caso contrário, imprimir “nao”. A saída deve ser impressa na saída padrão.

Exemplo de entrada:

Entrada:	Saída:
2 100 100 20 70 100 10 61 100 119 100 0	sim

Palíndromo Vocálico

Nome do arquivo fonte: pvocal.{c,cpp,java}

Um palíndromo é uma palavra ou número que quando lido de frente para trás ou de trás para frente é igual. Neste problema, vamos definir o conceito de palíndromo vocálico. Um palíndromo vocálico é uma palavra que quando lida de frente para trás é igual a ler de trás para frente, assumindo que todas as vogais devem ser consideras como se fossem a mesma letra. Para completar, um palíndromo vocálico pode possuir um caractere especial * (asterisco), que é lido como se fosse qualquer outra letra.

Alguns exemplos de palíndromos vocálicos:

- ovo
- amo
- faca*
- -*
- aeiou
- a*abebiboeu

Dada uma palavra como entrada, composta exclusivamente de até 40 letras minúsculas (a-z) ou asterisco (*), sua tarefa é informar a quantidade mínima de caracteres que podem ser retirados da palavra para que ela se torne um palíndromo vocálico.

Como exemplo, a palavra “faca” se tornaria um palíndromo vocálico simplesmente removendo a letra f, portanto a resposta para “faca” seria 1.

Entrada:

A entrada inicia por um inteiro positivo T (menor que 10000) informando a quantidade de casos de teste. Cada caso de teste aparece em uma linha isolada na entrada e é uma palavra com 1 a 40 letras (a-z) ou asterisco(*). A entrada deve ser lida da entrada padrão.

Saída:

Para cada palavra lida na entrada, seu programa deve exibir a quantidade mínima de caracteres que devem ser retirados da palavra para que ela se torne palíndromo vocálico. Note que se a palavra já for um palíndromo vocálico, essa resposta seria 0(zero). A saída deve ser impressa na saída padrão.

Exemplo de entrada:	**Respectiva saída:**
5 ovo * abacate umapalavrabemgrandeassim um*inormtto	0 0 2 11 2

Numbers

Nome do arquivo fonte: numbers.{c,cpp,java}

Little Petya likes numbers a lot. He found that number 123 in base 16 consists of two digits: the first is 7 and the second is 11. So the sum of digits of 123 in base 16 is equal to 18. Now he wonders what is an average value of sum of digits of the number *A* written in all bases from 2 to *A* - 1.

Note that all computations should be done in base 10. You should find the result as an irreducible fraction, written in base 10.

Input:

Input contains various lines with one integer number *A* ($3 \leq A \leq 1000$). When A=0 stop.

Output:

Output should contain required average value in format "X/Y", where X is the numerator and Y is the denominator.

Samples:

Input:	**Output:**
5 3 0	7/3 2/1

Números

Nome do arquivo fonte: numbers.{c,cpp,java}

Little Petya gosta muito de números.Ele descobriu que o número 123 em base 16 consiste de dois dígitos: o primeiro é 7 eo segundo é 11. Assim, a soma dos algarismos de 123 em base 16 é igual a 18.

Agora, ele quer saber qual é um valor médio da soma dos algarismos do número A escrito em todas as bases de 2 a *A* - 1.

Note-se que todos os cálculos devem ser feitos na base 10.Você deve encontrar o resultado como uma fração irredutível, escrito em base 10.

Entrada:

Entrada contém várias linhas com um número inteiro *A* (A ≤ 3 ≤ 1000).Quando A = 0 parar.

Saída:

A saída deve conter o valor médio em formato "X / Y, onde X é o numerador e Y é o denominador.

Exemplos:

Entrada:	**saída:**
5 3 0	7/3 2/1

Correct solution?

Nome do arquivo fonte: correct.{c,cpp,java}

One cold winter evening Alice and her older brother Bob was sitting at home near the fireplace and giving each other interesting problems to solve. When it was Alice's turn, she told the number n to Bob and said:

—Shuffle the digits in this number in order to obtain the smallest possible number without leading zeroes.

—No problem! — said Bob and immediately gave her an answer.

Alice said a random number, so she doesn't know whether Bob's answer is correct. Help her to find this out, because impatient brother is waiting for the verdict.

Input:

The first line contains one integer T, number of test cases. All cases consist with a first line contains one integer n ($0 \le n \le 10 \hat{} 9$) without leading zeroes. The second lines contains one integer m ($0 \le m \le 10 \hat{} 9$) — Bob's answer, possibly with leading zeroes.

Output:

Print OK if Bob's answer is correct and WRONG_ANSWER otherwise.

Samples:

Input:	**Output:**
2 3310 1033 4 5	OK WRONG_ANSWER

Solução correta?

Nome do arquivo fonte: correct.{c,cpp,java}

Em uma fria noite de inverno, Alice e seu irmão mais velho Bob estavam em casa sentados perto da lareira, dando um ao outro problemas interessantes para se resolver. Quando chegou a vez de Alice, ela deu um número N a Bob e disse:

—Embaralhe os algorismos deste número, a fim de obter o menor número possível, sem zeros a esquerda.

—Sem problema! — disse Bob e imediatamente deu uma resposta.

Alice disse um número aleatório, de forma que ela não sabe se a resposta de seu irmão está correta. Ajude Alice a descobrir se a solução do seu irmão está correta, pois ele está ansioso a espera do veredito.

Entrada:

A primeira linha contém um inteiro T, o número de casos teste. Todos os casos consiste de uma primeira linha contendo um número inteiro N ($0 \leq N \leq 10^9$), sem zeros à esquerda. As segundas linhas contém um número inteiro M ($0 \leq M \leq 10^9$) — a resposta de Bob, que eventualmente pode conter zeros a esquerda.

Saída:

Imprimir OK se a resposta do Bob está correta e WRONG_ANSWER caso contrário.

Exemplos:

Entrada:	**Saída:**
2 3310 1033 4 5	OK WRONG_ANSWER

Alien Language

Nome do arquivo fonte: alien.{c,cpp,java}

After years of study, scientists at Google Labs have discovered an alien language transmitted from a faraway planet. The alien language is very unique in that every word consists of exactly L lowercase letters. Also, there are exactly D words in this language.

Once the dictionary of all the words in the alien language was built, the next breakthrough was to discover that the aliens have been transmitting messages to Earth for the past decade. Unfortunately, these signals are weakened due to the distance between our two planets and some of the words may be misinterpreted. In order to help them decipher these messages, the scientists have asked you to devise an algorithm that will determine the number of possible interpretations for a given pattern.

A pattern consists of exactly L tokens. Each token is either a single lowercase letter (the scientists are very sure that this is the letter) or a group of unique lowercase letters surrounded by parenthesis (and). For example: (ab)d(dc) means the first letter is either a or b, the second letter is definitely d and the last letter is either d or c. Therefore, the pattern (ab)d(dc) can stand for either one of these 4 possibilities: add, adc, bdd, bdc.

Input:

The first line of input contains 3 integers, L, D and N separated by a space. D lines follow, each containing one word of length L. These are the words that are known to exist in the alien language. N test cases then follow, each on its own line and each consisting of a pattern as described above. You may assume that all known words provided are unique.

$1 \leq L \leq 15$

$1 \leq D \leq 5000$

$1 \leq N \leq 500$

Output:

For each test case, output :

Case #X: K

where X is the test case number, starting from 1, and K indicates how many words in the alien language match the pattern.

Samples:

Input :	**Output :**
3 5 4 abc bca dac dbc cba (ab)(bc)(ca) abc (abc)(abc)(abc) (zyx)bc	Case #1: 2 Case #2: 1 Case #3: 3 Case #4: 0

Linguagem alienígena

Nome do arquivo fonte: alien.{c,cpp,java}

Depois de anos de estudo, os cientistas no Google Labs descobriu uma língua alienígena transmitida de um planeta distante. A língua alienígena é muito singular, cada palavra é composta de exatamente L letras minúsculas. Além disso, existem exatamente D palavras neste idioma.

Uma vez que o dicionário de todas as palavras do idioma estrangeiro foi construído, o próximo passo foi descobrir o que os alienígenas estavam transmitindo nas mensagens à Terra na última década. Infelizmente estes sinais são enfraquecidos devido a distância entre os planetas e algumas das palavras podem ser mal interpretadas. A fim de ajudá-los a decifrar estas mensagens, os cientistas pediram-lhe para elaborar um algoritmo que determina o número de interpretações possíveis para um determinado padrão.

A estrutura consiste de exatamente L marcadores. Cada marcador pode ser uma letra minúscula única (neste caso os cientistas estão certos de que este marcador é a letra correta), ou um conjunto único de letras minúsculas entre parênteses. Por exemplo (ab)d(dc) significa que a primeira letra pode ser a ou b, a segunda letra é definitivamente d, e a terceira letra pode ser d ou c. Portanto, o padrão (ab)d(dc) pode representar qualquer uma dessas quatro possibilidades: add, adc, bdd, bdc.

Entrada:

A primeira linha da entrada contém três números inteiros L, D e N separados por um espaço. As próximas D linhas contém uma palavra de comprimento L. Estas são as palavras que são conhecidas na língua estrangeira. Após a última palavra, N casos de testes se seguem, cada um em sua própria linha, escritos na forma do padrão descrito acima. Você pode assumir que todas as palavras conhecidas fornecidas são únicas.

$1 \leq L \leq 15$

$1 \leq D \leq 5000$

$1 \leq N \leq 500$

Saída:

Para cada caso de teste a saída é:

Case #X: K

Onde X é o número corrente do caso de teste começando de 1, e K indica quantas palavras conhecidas na língua alienígena corresponde ao padrão.

Exemplos:

Entrada :	**Saída :**
3 5 4 abc bca dac dbc cba (ab)(bc)(ca) abc (abc)(abc)(abc) (zyx)bc	Case #1: 2 Case #2: 1 Case #3: 3 Case #4: 0

The House of Santa Claus

Nome do arquivo fonte: santa.{c,cpp,java}

In your childhood you most likely had to solve the riddle of the house of Santa Claus. Do you remember that the importance was on drawing the house in a stretch without lifting the pencil and not drawing a line twice? As a reminder it has to look like shown in Figure 1.

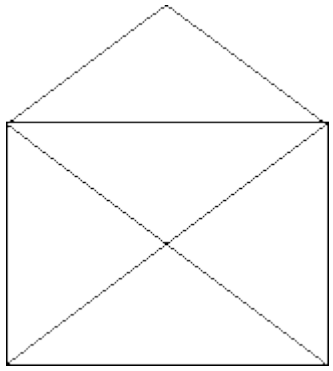


Figure: The House of Santa Claus

Well, a couple of years later, like now, you have to ``draw'' the house again but on the computer. As one possibility is not enough, we require *all* the possibilities when starting in the lower left corner. Follow the example in Figure 2 while defining your stretch.

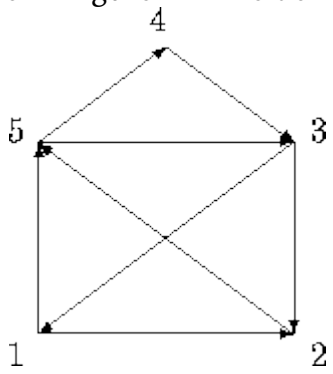


Figure: This Sequence would give the Outputline 153125432

All the possibilities have to be listed in the outputfile by increasing order, meaning that 1234... is listed before 1235... .

Input :
não tem

Output :
So, an outputfile could look like this:
12435123
13245123
...
15123421

A casa do Papai Noel

Nome do arquivo fonte: santa.{c,cpp,java}

Em sua infância, você provavelmente teve que resolver o enigma da casa do Papai Noel. Você se lembra que era importante desenhar a casa sem levantar o lápis do papel e sem desenhar a mesma linha duas vezes? Para relembrar veja a Figura 1.

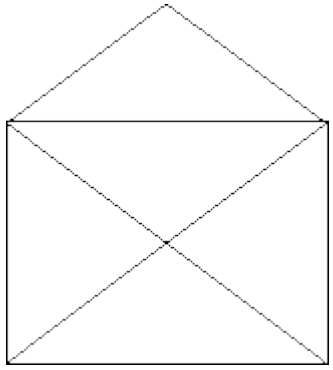


Figure 1: A casa do Papai Noel

Bem, alguns anos mais tarde, como agora, você tem que “desenhar” a casa novamente, porém no computador. Como uma possibilidade não é suficiente, precisamos de todas as possibilidades quando se inicia o desenho pelo canto inferior esquerdo. Siga o exemplo da figura 2. Enquanto define seu caminho.

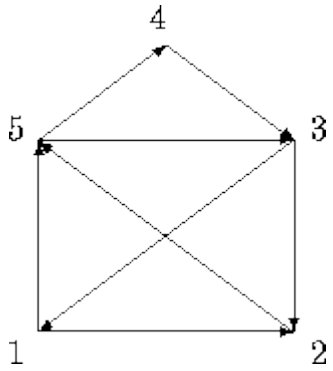


Figure: Esta sequência daria a saída 153125432

Todas as possibilidades têm de ser listadas na saída por ordem crescente, ou seja, 1234 ... é listado antes de 1235 ... .

Entrada :
não tem!

Saída :
12435123
13245123
...
15123421

Convex Hull Finding

Nome do arquivo fonte: convex.{c,cpp,java}

Given a single connected contour, which is either convex or non-convex (concave), use any algorithm to find its Convex Hull, i.e., the smallest convex contour enclosing the given shape. If the given contour is convex, then its convex hull is the original contour itself. The maximal size of the shape is 512x512, and the maximal number of the vertices of the shape is 512. Write a program to read the input data (the given shapes) from a disk file, implement your convex hull finding algorithm, and then output the shape data of the results to the standard output.

Input:

The order of the vertices is counterclockwise in *X-Y* Cartesian Plane (if you consider the origin of the display window is on the upper-left corner, then the orientation of the vertices is clockwise), and none of the neighboring vertices are co-linear. Since all the shapes are closed contours, therefore, the last vertex should be identical to the first vertex. There are several sets of data within a given data file. The negative number -1 is used to separate the data set.

Line Number	Data in the File	Explanation
1	*K*	a positive integer showing how many sets of data in this file
2	*N*	a positive integer showing the number of vertices for the shape
3	X_1 Y_1	two positive integers for the first vertex (X_1, Y_1)
4	X_2 Y_2	two positive integers for the next neighboring vertex (X_2, Y_2)
...		
N+2	X_N Y_N	two positive integers for the last vertex (X_N, Y_N)
N+3	-1	Delimiter
N+4	*M*	a positive integer showing the number of vertices for the next shape
N+5	XX_1 YY_1	two positive integers for the first vertex
...		

Note: Please note that the Line Number, Data in the File and Explanation are not given in the file. They are shown here only to assist you in reading the data.

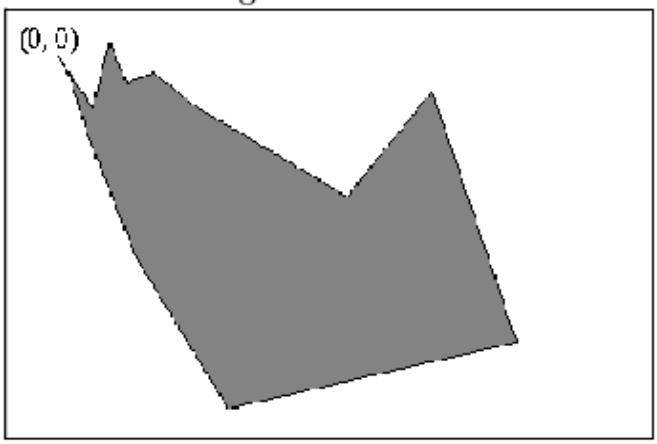
Output:

Output the convex hull of all *K* input shapes to the standard output. The data format should be the same as the input file. In addition, the vertex with the smallest *Y* value should be the first point and if there are points with the same *Y* value, then the smallest *X* value within those points should be the first point.

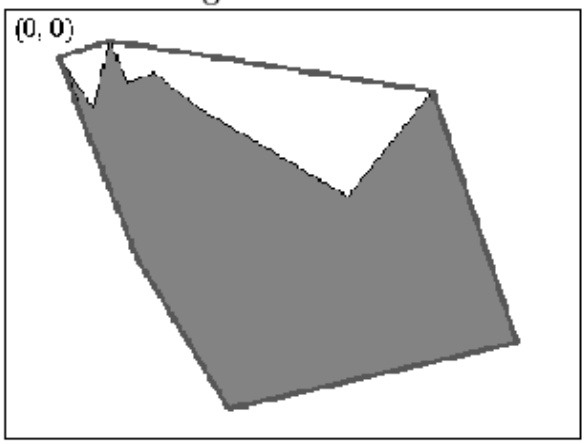
Samples:

Input :	**Output :**
3 15 30 30 50 60 60 20 70 45 86 39 112 60 200 113 250 50 300 200 130 240 76 150 47 76 36 40 33 35 30 30 -1 12 50 60 60 20 70 45 100 70 125 90 200 113 250 140 180 170 105 140 79 140 60 85 50 60 -1 6 60 20 250 140 180 170 79 140 50 60 60 20	3 8 60 20 250 50 300 200 130 240 76 150 47 76 30 30 60 20 -1 6 60 20 250 140 180 170 79 140 50 60 60 20 -1 6 60 20 250 140 180 170 79 140 50 60 60 20

The contour shape of the first data
set is shown in figure as follows:



The convex hull of the above shape is shown in the following figure:



Procurando o Fecho Convexo

Nome do arquivo fonte: convex.{c,cpp,java}

Dado um contorno único ligado, que seja convexo ou não-convexo (côncavo), use um algoritmo para encontrar o seu Fecho Convexo, ou seja, o menor contorno convexo envolvendo a forma determinada. Se o contorno convexo é dado, então o seu fecho convexo é o contorno original próprio. O tamanho máximo da forma é 512x512 e o número máximo de vértices da forma é 512. Escreva um programa para ler os dados de entrada (as formas dadas) a partir de um arquivo em disco, implemente o seu algoritmo de encontrar o fecho convexo e a saída de dados segundo a forma dos resultados para a saída padrão.

Entrada:

A ordem dos vértices é sentido anti-horário no Plano Cartesiano X-Y (se você considerar a origem da janela de visualização está no canto superior esquerdo, em seguida, a orientação dos vértices é no sentido horário), e nenhum dos vértices vizinhos são co-lineares. Como todas as formas são contornos fechados, portanto, o último vértice deve ser idêntico ao primeiro vértice. Há vários conjuntos de dados dentro de um determinado arquivo de dados. O número negativo -1 é usado para separar o conjunto de dados.

Line Number	Data in the File	Explanation
1	K	a positive integer showing how many sets of data in this file
2	N	a positive integer showing the number of vertices for the shape
3	$X_1\ Y_1$	two positive integers for the first vertex (X_1, Y_1)
4	$X_2\ Y_2$	two positive integers for the next neighboring vertex (X_2, Y_2)
...		
N+2	$X_N\ Y_N$	two positive integers for the last vertex (X_N, Y_N)
N+3	-1	Delimiter
N+4	M	a positive integer showing the number of vertices for the next shape
N+5	$XX_1\ YY_1$	two positive integers for the first vertex
...		

Nota: Repare que o número da linha, os dados no arquivo e explicação não é dado no arquivo. Eles são mostrados aqui apenas para ajudar na leitura dos dados.

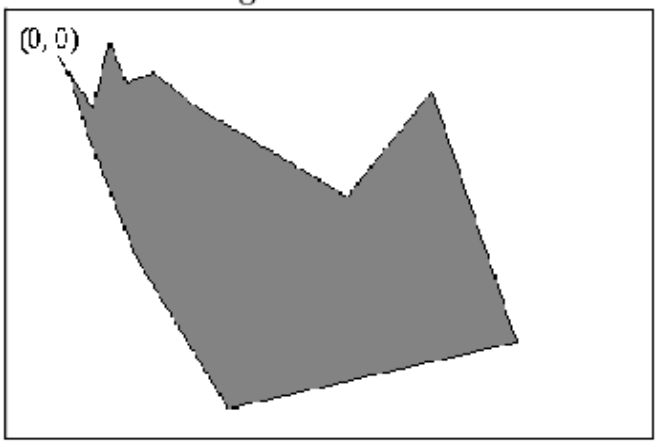
Saída:

Saída do fecho convexo de todas as K formas de entrada para a saída padrão. O formato dos dados deve ser o mesmo que o arquivo de entrada. Além disso, o vértice com o menor valor de Y deve ser o primeiro ponto e, se há pontos com o valores iguais de Y, então o menor valor de X dentro desses pontos deve ser o primeiro ponto.

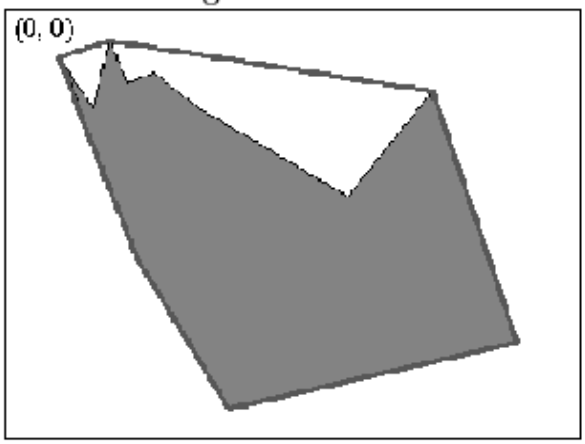
Samples:

Input :	**Output :**
3 15 30 30 50 60 60 20 70 45 86 39 112 60 200 113 250 50 300 200 130 240 76 150 47 76 36 40 33 35 30 30 -1 12 50 60 60 20 70 45 100 70 125 90 200 113 250 140 180 170 105 140 79 140 60 85 50 60 -1 6 60 20 250 140 180 170 79 140 50 60 60 20	3 8 60 20 250 50 300 200 130 240 76 150 47 76 30 30 60 20 -1 6 60 20 250 140 180 170 79 140 50 60 60 20 -1 6 60 20 250 140 180 170 79 140 50 60 60 20

The contour shape of the first data
set is shown in figure as follows:



The convex hull of the above shape is shown in the following figure:



What is the median?

Nome do arquivo fonte: median.{c,cpp,java}

Median plays an important role in the world of statistics. By definition, it is a value which divides an array into two equal parts. In this problem you are to determine the current median of some long integers.

Suppose, we have five numbers {1,3,6,2,7}. In this case, 3 is the median as it has exactly two numbers on its each side. {1,2} and {6,7}.

If there are even number of values like {1,3,6,2,7,8}, only one value cannot split this array into equal two parts, so we consider the average of the middle values {3,6}. Thus, the median will be (3+6)/2 = 4.5. In this problem, you have to print only the integer part, not the fractional. As a result, according to this problem, the median will be 4!

Input:

The input file consists of series of integers X (0 <= X < 2 ^ 31) and total number of integers N is less than 10000. The numbers may have leading or trailing spaces.

Output:

For each input print the current value of the median.

Samples:

Input :	**Output :**
1 3 4 60 70 50 2	1 2 3 3 4 27 4

Qual é a média?

Nome do arquivo fonte: median.{c,cpp,java}

Mediana tem um papel importante no mundo das estatísticas. Por definição, é um valor que divide um array em duas partes iguais. Neste problema você deve determinar a média atual de alguns inteiros longos.
Suponha que nós temos cinco números {1,3,6,2,7}. Neste caso, 3 é a mediana, uma vez que tem exatamente dois números do seu lado cada um. {1,2} e {6,7}.
Se houver números pares de valores como {1,3,6,2,7,8}, apenas um valor não pode dividir essa matriz em duas partes iguais, por isso, considerar a média dos valores médios (3,6). Assim, a mediana será (3 +6) / 2 = 4.5. Neste problema, você tem que imprimir somente a parte inteira, e não o fracionário. Como resultado, de acordo com este problema, a mediana será de 4!

Entrada:

O arquivo de entrada consiste na serie de inteiros X (0 <= X < 2^31) e o número total de inteiros N é menor que 10000. Os números podem ter um espaço em branco na frente ou atrás.

Output:

Para cada entrada imprimir o valor atual da mediana.

Samples:

Input :	**Output :**
1 3 4 60 70 50 2	1 2 3 3 4 27 4

Generalized Matrioshkas

Nome do arquivo fonte: matriosh.{c,cpp,java}

Vladimir worked for years making *matrioshkas*, those nesting dolls that certainly represent truly Russian craft. A matrioshka is a doll that may be opened in two halves, so that one finds another doll inside. Then this doll may be opened to find another one inside it. This can be repeated several times, till a final doll -that cannot be opened- is reached.

Recently, Vladimir realized that the idea of nesting dolls might be generalized to nesting toys. Indeed, he has designed toys that contain toys but in a more general sense. One of these toys may be opened in two halves and it may have more than one toy inside it. That is the new feature that Vladimir wants to introduce in his new line of toys.

Vladimir has developed a notation to describe how nesting toys should be constructed. A toy is represented with a positive integer, according to its size. More precisely: if when opening the toy represented by $\boldsymbol{m}$ we find the toys represented by $\boldsymbol{n}_1$, $\boldsymbol{n}_2$, ..., $\boldsymbol{n}_r$, it must be true that $\boldsymbol{n}_1 + \boldsymbol{n}_2 + ... + \boldsymbol{n}_r < \boldsymbol{m}$. And if this is the case, we say that toy $\boldsymbol{m}$ contains directly the toys $\boldsymbol{n}_1$, $\boldsymbol{n}_2$, ..., $\boldsymbol{n}_r$. It should be clear that toys that may be contained in any of the toys $\boldsymbol{n}_1$, $\boldsymbol{n}_2$, ..., $\boldsymbol{n}_r$ are not considered as directly contained in the toy $\boldsymbol{m}$.

A *generalized matrioshka* is denoted with a non-empty sequence of non zero integers of the form:

$$a_1 \quad a_2 \quad ... \quad a_N$$

such that toy $\boldsymbol{k}$ is represented in the sequence with two integers - $\boldsymbol{k}$ and $\boldsymbol{k}$, with the negative one occurring in the sequence first that the positive one.

For example, the sequence

-9 -7 -2 2 -3 -2 -1 1 2 3 7 9

represents a generalized matrioshka conformed by six toys, namely, 1, 2 (twice), 3, 7 and 9. Note that toy7 contains directly toys 2 and 3. Note that the first copy of toy 2 occurs left from the second one and that the second copy contains directly a toy 1. It would be wrong to understand that the first -2 and the last 2should be paired.

On the other hand, the following sequences do not describe generalized matrioshkas:

-9 -7 -2 2 -3 -1 -2 2 1 3 7 9

because toy 2 is bigger than toy 1 and cannot be allocated inside it.

-9 -7 -2 2 -3 -2 -1 1 2 3 7 -2 2 9

because 7 and 2 may not be allocated together inside 9.

-9 -7 -2 2 -3 -1 -2 3 2 1 7 9

because there is a nesting problem within toy 3.

Your problem is to write a program to help Vladimir telling good designs from bad ones.

Input:

The input file contains several test cases, each one of them in a separate line. Each test case is a sequence of non zero integers, each one with an absolute value less than 10^7.

Output:

Output texts for each input case are presented in the same order that input is read.

For each test case the answer must be a line of the form

:-) Matrioshka!

if the design describes a generalized matrioshka. In other case, the answer should be of the form

:-(Try again.

Samples:

Input :	**Output :**
-9 -7 -2 2 -3 -2 -1 1 2 3 7 9	:-) Matrioshka!
-9 -7 -2 2 -3 -1 -2 2 1 3 7 9	:-(Try again.
-9 -7 -2 2 -3 -1 -2 3 2 1 7 9	:-(Try again.
-100 -50 -6 6 50 100	:-) Matrioshka!
-100 -50 -6 6 45 100	:-(Try again.
-10 -5 -2 2 5 -4 -3 3 4 10	:-) Matrioshka!
-9 -5 -2 2 5 -4 -3 3 4 9	:-(Try again.

Matrioshkas Generalizadas

Nome do arquivo fonte: matriosh.{c,cpp,java}

Vladimir trabalhou por muitos anos fazendo matrioshkas, aquelas bonecas que certamente representam verdadeiramente o artesanato russo. A matrioshka é uma boneca que pode ser aberta em duas metades, de modo que se encontra dentro de outra boneca. Então, esta boneca pode ser aberta para encontrar uma outra dentro dela. Isso pode ser repetido várias vezes, até que a boneca final, que não pode ser aberta é alcançada.

Recentemente, Vladimir percebeu que a idéia das bonecas pode ser generalizada para aninhar brinquedos. Na verdade, ele criou os brinquedos que contêm brinquedos, mas num sentido mais geral. Um desses brinquedos podem ser aberto em duas metades e pode ter mais um brinquedo dentro. Essa é a nova funcionalidade que Vladimir pretende introduzir na sua nova linha de brinquedos.
Vladimir desenvolveu a notação para descrever como o aninhamento de brinquedos deve ser construído. Um brinquedo é representado com um número inteiro positivo, de acordo com seu tamanho. Mais precisamente: se ao abrir o brinquedo representada por $\boldsymbol{m}$ encontramos os brinquedos representada por $\boldsymbol{n}_1$, $\boldsymbol{n}_2$, ..., $\boldsymbol{n}_r$, deve ser verdade que $\boldsymbol{n}_1 + \boldsymbol{n}_2 + ... + \boldsymbol{n}_r < \boldsymbol{m}$. E se esse for o caso, dizemos que o brinquedo $\boldsymbol{m}$ contém diretamente os brinquedos $\boldsymbol{n}_1$, $\boldsymbol{n}_2$, ..., $\boldsymbol{n}_r$ .Deve ficar claro que os brinquedos que podem estar contidos em qualquer dos brinquedos $\boldsymbol{n}_1$, $\boldsymbol{n}_2$, ..., $\boldsymbol{n}_r$ não são considerados como contidos diretamente no brinquedo $\boldsymbol{m}$.
Uma matrioshka generalizada é denotada com uma sequência não vazia de inteiros diferentes de zero da seguinte forma:

$$a_1 \quad a_2 \quad ... \quad a_N$$

cada brinquedo $\boldsymbol{k}$ é representado por uma sequência de dois inteiros - $\boldsymbol{k}$ e $\boldsymbol{k}$, com o negativo one occurring in the sequence first that the positive one.
Por exemplo, a seqüência

-9 -7 -2 2 -3 -2 -1 1 2 3 7 9

representa uma matrioshka generalizada formada por seis brinquedos, ou seja, 1, 2 (dois), 3, 7 e 9. Note-se que brinquedo7 contém diretamente brinquedos 2 e 3. Observe que o primeiro exemplar do brinquedo 2 ocorre à esquerda da segunda e que a segunda cópia contém diretamente um brinquedo 1. Seria errado compreender que o primeiro -2 e o último 2 possam ser emparelhados.
Por outro lado, as seguintes seqüências não descreve matrioshkas generalizada:

-9 -7 -2 2 -3 -1 -2 2 1 3 7 9

porque brinquedo 2 é maior do que o brinquedo um e não podem ser alocado dentro dele.

-9 -7 -2 2 -3 -2 -1 1 2 3 7 -2 2 9

porque 7 e 2 não podem ser atribuídas em conjunto dentro de 9

-9 -7 -2 2 -3 -1 -2 3 2 1 7 9

porque há um problema de aninhamento dentro do brinquedo 3.

Seu problema é escrever um programa para ajudar Vladimir dizendo que desenhos são bons ou ruins.

Entrada:

O arquivo de entrada contém vários casos de teste, cada um deles em uma linha separada. Cada caso de teste é uma sequência de não-zeros inteiros, cada uma com um valor absoluto inferior a 107.
Saída:
textos de saída para cada caso de entrada são apresentados na mesma ordem que a entrada é lida.
Para cada caso de teste, a resposta deve ser uma linha da forma

:-) Matrioshka!

se o projeto descreve uma matrioshka generalizada. Em outro caso, a resposta deve ser da forma

:-(Try again.

Samples:

Input :	**Output :**
-9 -7 -2 2 -3 -2 -1 1 2 3 7 9	:-) Matrioshka!
-9 -7 -2 2 -3 -1 -2 2 1 3 7 9	:-(Try again.
-9 -7 -2 2 -3 -1 -2 3 2 1 7 9	:-(Try again.
-100 -50 -6 6 50 100	:-) Matrioshka!
-100 -50 -6 6 45 100	:-(Try again.
-10 -5 -2 2 5 -4 -3 3 4 10	:-) Matrioshka!
-9 -5 -2 2 5 -4 -3 3 4 9	:-(Try again.