Problema A

A fórmula de Euler

Nome do arquivo fonte: **euler.c, euler.cpp ou euler.java**

Leonhard Paul Euler (Basileia, 15 de abril de 1707 — São Petersburgo, 18 de setembro de 1783) foi um matemático e físico suíço de língua alemã que passou a maior parte de sua vida na Rússia e na Alemanha.

Uma das grandes áreas que Euler contribuiu foi a geometria. O estudo dos poliedros está frequentemente ligado ao problema da medição de certas grandezas (volume, área das faces, comprimento das arestas, amplitude dos ângulos diedrais, ...). Mas os poliedros podem ser também interessantes de outro ponto de vista: uma igualdade descoberta por Euler em 1751 relaciona os números V de vértices, F de faces e A de arestas.

Alguns dos rascunhos de Euler para chegar a conclusão da relação entre V, F e A foram achados recentementes e alguns estão incompletos e outros não. O seu trabalho é pegar essas dados e verificar as anotações de Euler.

Entrada

A primeira linha indica o número de casos de teste, N ($1 \leq N \leq 10000$). As N linhas seguintes são os rascunhos a serem verificados. Cada linha consiste de uma permutação de valores de V, F e A, por exemplo:

1. V=<valor> A = <valor> F=<valor>
2. A=<valor> V=<valor> F=<valor>

Outras permutações são possíveis e o valor pode ser igual a "?" ou a um valor inteiro positivo com até maior que 0 zero e menor ou igual à 10000

Saída

A saída vai ser a verificação de cada uma das N linhas da entrada separadas por uma quebra de linha. Se todos os dados da entrada estiverem completos seu programa deve imprimir "Correto" caso sejam válidos de acordo com a Fórmula de Euler e "Impossível" caso contrário. No caso de haver dados incompletos("?"), o seu programa deve imprimir uma solução caso exista para a entrada com o seguinte formato, por exemplo: V=10; caso contrário imprimir "Impossivel".

Exemplo:

Entrada	Saída
3 V=4 F=4 A=? V=? F=? A=? A=6 F=4 V=4	A=6 Impossivel Correto

Dicas:

i) $A \geq 6$
ii) $V - A + F = 2$
iii) $A + 6 \leq 3F \leq 2A$
iv) $A + 6 \leq 3V \leq 2A$

Problema B

Informa

Nome do arquivo fonte: **informa.c, informa.cpp ou informa.java**

Os pontos de entrega do Informa consistem da intersecção de caminhos que os conectam.

Todos as segundas-feira o serviço de entrega do Jornal Informa da UFES precisa entregar os jornais nos seus pontos de entrega. Todos os pontos de entrega são conectados a pelo menos um ponto de entrega e existe apenas um caminho entre cada ponto de entrega. Durante a jornada de trabalho a UFES tem a disposição dois funcionários para a entrega, Severino e Juvenal, que começam sempre do mesmo ponto a jornada.

Buscando uma eficiência na entrega a UFES delegou a vocês, criar um programa capaz de receber todos os pontos de entrega e o ponto de partida e calcular o custo da rota de menor custo total. Sabe-se que Severino e Juvenal estão dispensados do dia de trabalho assim que acabam a rotinha de entrega e que não necessariamente terminam a rotina no mesmo ponto de entrega.

Entrada

Cada entrada consiste de uma linha contendo dois inteiros: N e S, $1 \leq N \leq 100000$, $1 \leq S \leq N$. N é número total de intersecções; S é o número ordinal do ponto de partida. Os pontos de intersecção são marcados por números 1, 2, 3, …, N.

As N-1 linhas seguintes contém 3 inteiros: A, B e C, onde A e B são duas intersecções conectadas com um custo C, $1 \leq C \leq 1000$.

O programa dever encerrar sua rotina quando N e S forem iguais a 0 (zero)

Saída

Seu programa deve exibir o custo total da rota separados por linha.

Exemplo:

Entrada	Saída
5 2 1 2 1 2 3 2 3 4 2 4 5 1 5 1 1 2 1 2 3 1 3 5 1 3 4 1 4 1 1 3 2	6 5 11

1 2 3 1 4 4 0 0	

Problema C

Colônia de formigas

Nome do arquivo fonte: **formigas.c, formigas.cpp ou formigas.java**

Uma colônia de formigas é formada por diferentes tipos de indivíduos adultos. Cada um desses tipos constitui uma casta. Sendo assim, uma colônia de formigas é formada numa casta reprodutiva (formada pela rainha e pelos machos) e por operárias que podem se dividir em sub-castas (soldados).

Revoltado com a quantidade de colônias de formigas em sua fazenda, sabendo-se que você tem apenas material suficiente para eliminar duas colônias, ou seja, as duas maiores. Faça um programa capaz de dizer qual são essas duas colônias.

Entrada

O número total de colônias N ($2 \leq N \leq 100000$) e as N linhas seguintes o tamanho da população de formigas P ($1 \leq P \leq 10^{200}$).

Saída

As duas maiores colônias separadas por linha em ordem crescente.

Exemplo:

Entrada	Saída
10 1 2 3 4 5 6 7 8 9 10	9 10

Problema D

Bombas

Nome do arquivo fonte: **bombas.c, bombas.cpp ou bombas.java**

Todos conhecem o país nlogônia que, durante as competições do ICPC, vem sofrido ataques do mais diversos (como o Ataque Fulminante da Primeira Fase da Maratona de Programação 2009). O país nlogônia, como forma de contra-ataque, está prestes a lançar duas bombas em território inimigo. As bombas ao caírem destoem uma região em forma de circunferência e, por problemas de logística de Nlogônia, as bombas cairão em regiões próximas (correndo o risco de caírem na mesma posição), diminuindo a área destruída. Como aliado, você deve dizer qual será a área da cidade destruída, dado a posição e o raio de alcance de cada uma das duas bombas lançada.

Entrada

A entrada é composta por vários casos de teste. Cada caso de teste representa a localização e o raio de alcançe de ambas as bombas, no formato x_1, y_1, r_1, x_2, y_2 e r_2 , sendo todos os valores inteiros. A entrada termina com o final do arquivo.

$0 \leq |x_1|, |x_2|, |y_1|, |y_2| \leq 1000$
$1 \leq r_1, r_2 \leq 100$

Saída

Para cada caso de teste, seu programa deve imprimir três linhas. A primeira linha deve conter “Teste n”, onde n representa a instância de teste (1, 2, 3, ….). A segunda linha deve conter um número que represente a área da cidade destruída, com precisão de 3 casas decimais e a terceira linha deve ser deixada em branco.

Exemplo:

Entrada	Saída
1 2 3 1 2 2 0 0 2 1 0 1 0 0 2 2 0 2 0 0 1 2 0 1 0 0 2 100 0 99	Teste 1 28.274 Teste 2 12.566 Teste 3 20.219 Teste 4 6.283

	Teste 5 30800.877

Problema E

Cálculo do determinante

Nome do arquivo fonte: **matriz.c, matriz.cpp ou matriz.java**

Dada uma matriz A com o seguinte formato:

$$\underbrace{\begin{vmatrix} b^2+1 & b & 0 & 0 & \dots & 0 & 0 \\ b & b^2+1 & b & 0 & \dots & 0 & 0 \\ 0 & b & b^2+1 & b & \dots & 0 & 0 \\ 0 & 0 & b & b^2+1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & b^2+1 & b \\ 0 & 0 & 0 & 0 & \dots & b & b^2+1 \end{vmatrix}}_{\text{n colunas}} \Big\} n \text{ linhas}$$

Onde, $b^2 \neq 1$.
Calcule o determinante de A dado o valor de n e b.

Entrada

Cada linha de teste contém os valores de n e b respectivamente. O programa deve encerrar a execução quando n e b forem iguais a 0 (zero).

Limites: $1 \leq n \leq 30, 1 < b \leq 30$.

Saída

A cada teste deve ser impresso o valor do determinante de A separados por linha.

Exemplo:

Entrada	Saída
4 5 5 4 0 0	406901 1118481

Problema F

Sequências de Röntgen

Nome do arquivo fonte: **rontgen.c, rontgen.cpp ou rontgen.java**

Wilhelm Conrad Röntgen foi um físico alemão que viveu no final do século XIX e início do século XX. Suas experiências em radiação eletromagnéticas renderam a ele o primeiro prêmio Nobel em Física, outorgado em 1901. Sua principal descoberta foi a existência do "raio X" e seu uso em aplicações médicas. Em 22 de dezembro de 1895, Röntgen fez um raio X da mão de sua esposa (com um anel em um dos dedos). A descoberta do raio X causou grande alvoroço na época e já em 1896 jornais europeus noticiavam a invenção e as grandes possibilidades de enxergar por dentro dos corpos sem a necessidade de cortá-los. A morte de Röntgen, causada por um certo tipo de câncer, é atribuída às radiações constantes a que esteve exposto durante suas pesquisas científicas.

Röntgen começou a desconfiar da existência de radiações invisíveis quando, nas suas pesquisas, era capaz de medir alterações consideráveis na fluorescência dos objetos quando colocados num tubo de Lenard que era submetido a uma corrente elétrica. Os estudos de Röntgen foram tão precisos que ele pôde inclusive gerar a seqüência que era observada no tubo de Lenard em cada instante de tempo. A fluorescência observada dependia da intensidade da corrente (X) e do tempo em que o tubo era submetido à corrente (Y). Röntgen percebeu que dada a primeira seqüência, a próxima podia ser obtida descrevendo os números da seqüência anterior. Por exemplo: se a primeira seqüência for 2 então a próxima é 12 (ou seja, a seqüência anterior é formada por "um 2"), a seguinte 1112 (ou seja, a seqüência anterior é formada por um 1 e um 2), 3112 (ou seja, a seqüência anterior é formada por três 1 e um 2), e assim por diante.

Além de um cientista brilhante, Röntgen era extremamente organizado. Ele guardava todos os registros de seus experimentos. Infelizmente, com o tempo algumas seqüências foram danificadas e outras perdidas. Sua tarefa é dada uma seqüência, determinar as próximas K seqüências do experimento.

Entrada

Cada instância é composta por uma linha contendo a primeira seqüência do experimento, formada por não mais de 1000 caracteres de 0 a 9, e o número K de seqüências que desejamos gerar ($1 \leq K \leq 50$), respectivamente.

Saída

Para cada instância, imprima a seqüência dada na entrada seguida de K linhas contendo as seqüências na ordem que foram geradas. As seqüências geradas não terão mais do que 2000000 caracteres.

Após cada instância imprima uma linha em branco.

Exemplo:

Entrada	Saída
3 2 5 99 3 000123 3	2 12 1112 3112 132112 1113122112 99 29 1219 11121119 000123 30111213 131031121113 111311101321123113

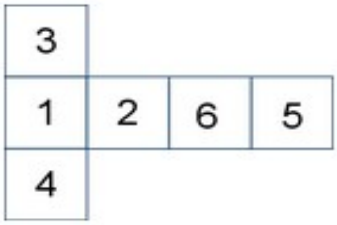
Problema G

Labirinto de dados

Nome do arquivo fonte: **labirinto.c, labirinto.cpp ou labirinto.java**

Renan Manolo é um cara concurseiro e, para parar de só estudar para concursos, ele resolveu começar a brincar com jogos de tabuleiros. Só que Manolo ficou enjoado desses jogos clássicos, então ele resolveu criar seu próprio jogo.

O Jogo consiste em um labirinto, onde você possui uma posição inicial(origem) e uma posição final(destino) que deseja chegar. Isso seria um problema fácil de resolver. Porém, para dificultar e deixar o problema mais divertido, Manolo utiliza como pedra de tabuleiro um DADO de 6 faces com as faces numeradas de 1 (zero) a 6 (seis), que pode ser representado como a figura abaixo:



O jogo funciona da seguinte maneira. Ao mover o DADO para um determinado lado, é necessário rola-lo durante o movimento. Então se a face 1 estiver para cima e o DADO for movimentado para a esquerda, a face 2 passará a ficar na parte de superior. Se em seguida for movimentado para a casa de baixo do tabuleiro, a casa 6 do DADO ficará na parte superior, e assim sucessivamente.

O jogo começa com o dado em uma determinada posição do labirinto e com a face 1 virada para cima (com a face direita no 2 e face esquerda no 5). O objetivo é movimentar o dado para uma posição final específica e, como critério, a face superior do dado deve ser um valor específico (definido no jogo).

Você foi encarregado de desenvolver um programa que, dado as configurações do jogo, diga qual a quantidade mínima de movimentos que Manolo precisará para movimentar o dado da posição inicial para a posição final, mantendo as regras do jogo terminar com uma face específica do DADO virada para cima.

Entrada

A entrada é composta por vários casos de teste. Cada caso de teste representa uma formação do tabuleiro. A primeira linha contem dois inteiros H e W ($2 \leq H, W \leq 100$) representando a altura e a largura do tabuleiro. As H linhas seguintes contém cada uma cadeia com W caracteres, onde um caractere . (ponto) representa uma posição do labirinto que é possível caminhar e um caractere # representa que existe um obstáculo nessa determinada posição, ou seja, que não é possível caminhar. A última linha é composta por 5 inteiros X_i, Y_i, X_f, Y_f e V_f ($1 \leq X_i, X_f \leq W, 1 \leq Y_i, Y_f \leq H e 1 \leq V_f \leq 6$) que representam a posicão da partida e da chegada no jogo, e o valor do dado (face superior) desejado na casa de destino. O final do arquivo é marcado

com H = W = 0.

Saída

A saida é composta de três linhas. A primeira linha contém “Teste n”, onde n é inteiro representa a instância do caso de teste. A segunda linha deve ter um número inteiro R que representa a quantidade mínima de movimentos necessárias para chegar da posição de origem até a posição de destino ou “Impossivel” caso não exista caminho que satisfaça os critérios do jogo. A terceira linha de cada caso de teste deve ser deixada em branco.

Exemplo:

Entrada	Saída
2 2 .. .. 1 1 2 2 1 5 10 .#..... .#.####.#. .#.#.....# ...###..#. .#...#.... 1 1 5 10 5 1 3 .#. 1 1 1 3 2 1 5 ..... 1 1 1 5 3 0 0	Teste 1 6 Teste 2 19 Teste 3 Impossivel Teste 4 Impossivel

Problema H

O robô do Hans

Nome do arquivo fonte: **robo.c, robo.cpp ou robo.java**

O mundo inteiro ficou obcecado com robôs, e para manter o ritmo com o progresso, o grande programador Hans decidiu construir seu próprio robô. Ele estava trabalhando duro no projeto do seu robô. Ele ensinou seu robô a percorrer o caminho mais curto de um ponto a outro, para gravar todos os seus movimentos, mas como em muitos outros programas de Hans, houve um erro - o robô não percorre sempre o caminho mais curto. Felizmente, o robô registra seus próprios movimentos corretamente. Agora Draude quer descobrir quando as seu robô funciona de maneira errada. Heh, se Hans apenas se lembra do mapa do campo, onde ele testou o robô, ele poderia facilmente dizer se o robô caminhava na direção certa ou não. Mas o mapa de campo foi perdido para nunca ser encontrado, por isso ele pede para você descobrir se existe pelo menos um mapa, onde o caminho gravado pelo robô é o mais curto.

O mapa é um campo infinito de xadrez, onde cada quadrado está vazio, ou contém uma obstrução. Sabe-se também que o robô nunca tenta percorrer através da obstrução. Pelos movimentos registrados do robô, descobrir se existe pelo menos um mapa tal que é possível escolher para que o robô de um quadrado de partida (o quadrado inicial deve estar vazia) de tal forma que quando o robô se move a partir desta praça seus movimentos coincidem com as os gravados (o robô não funciona em nada, movendo apenas os quadrados vazios), e o caminho da partida para o final é o mais curto.

Em um movimento que o robô pode se mover para o quadrado (desde que não hajam obstrutions nesta praça) que tem partes comuns com o quadro que o robô está dentro

Entrada

A entrada consiste de sucessivas linhas com no máximo 100 caracteres indicando os movimentos do robô.

Os movimentos são: U, D, L e R que significam subir, descer, esquerda e direita respectivamente.

Saída

Para cada entrada imprimir uma linha contendo OK se o caminho existe e BUG caso contrário.

Exemplo:

Entrada	Saída
LLUUUR RRUULLDD	OK BUG

Problema I

Oráculo de Alexandria

Nome do arquivo fonte: **oraculo.c, oraculo.cpp ou oraculo.java**

Todo computólogo que se preza conhece o livro "O guia do mochileiro das galáxias" (The Hit-chhiker's Guide to the Galaxy) e sabe qual é a resposta para a pergunta fundamental sobre a vida, o universo e tudo mais1 . Mas, o que poucos sabem, é que a história de Douglas Adams é baseada em uma lenda egípcia, de um oráculo situado na cidade de Eskendereyya (Alexandria). Alexandria hoje é a maior cidade do Egito, com mais de 4 milhões de habitantes. Fica no delta do Nilo, e extende-se por 32km na costa do Mediterrâneo. Na Antiguidade, a cidade fundada em 331 a.C. por Alexandre, o Grande, foi umas das principais cidades do mundo e lá ficava o Farol de Alexandria (uma das 7 maravilhas do mundo antigo), a Biblioteca de Alexandria (a maior do mundo antigo) além de outras obras fantásticas. A lenda diz também que lá ficava o grande oráculo de Alexandria. Os habitantes da cidade entregavam ao oráculo pequenos bilhetes com números anotados, e recebia de volta um número, que seria a resposta a uma pergunta fundamental do universo relacionada aos dois números dados.

No seu tratado de 227 d.C. Cleómenes de Naucratis (que se tornou administrador de Alexandria quando Alexandre partiu para suas conquistas) relata alguns resultados obtidos do oráculo:

- Dados 8 e 1 o oráculo devolvia 5040;
- Dados 10 e 3, devolvia 280;
- Dados 4 e 2, devolvia 8;
- Dados 21 e 19, devolvia 42.

Estudos modernos dão conta que o que o oráculo devolvia nada mais era que uma generalização do fatorial de um número inteiro. Como sabemos,

$$N\,! = N \times (N - 1) \times \ldots \times 1.$$

O oráculo devolvia para os dados N e K o K-fatorial de N , ou seja,

$$N \times (N - K) \times (N - 2K) \times (N - 3K) \times \ldots,$$

em que o produto era feito enquanto a diferença é maior ou igual a 1. Podemos representar o K-fatorial de um número por ele seguido por K exclamações:

- 7! = 5040;
- 10!!! = 280;
- 4!! = 8;
- 21!!!!!!!!!!!!!!!!!! = 42

Dizem que ao ler sobre a lenda do oráculo de Eskendereyya, Douglas Adams teve sua inspiração para sua obra. Também, no Egito está a inspiração do Restaurante do fim do universo, mas isso é outra história... Sua tarefa é dado inteiros N e K determinar K-fatorial de N .

Entrada

A entrada é composta por diversas instâncias. A primeira linha da entrada contém um inteiro T

indicando o número de instâncias.
A primeira (e única) linha de cada instância contém um inteiro N seguido de K pontos de exclamação, onde $1 \leq N \leq 100$ e $1 \leq K \leq 20$.

Saída

Para cada instância imprima uma linha contendo o K-fatorial de N .
É garantido que nenhuma instância na entrada possui resultado maior que 10^{18} .

Exemplo:

Entrada	Saída
4 3! 10!!! 19!!!! 4!!	6 280 65835 8

Problema J

Contando ciclos

Nome do arquivo fonte: **task.c, task.cpp ou task.java**

Dado um grafo simples, imprimir a quantidade de ciclos do grafo. Um ciclo simples é um ciclo sem repetição de vertices ou aresta.

Entrada

A primeira linha de cada entrada contém dois inteiros n e m ($1 \leqslant n \leqslant 19, m \geqslant 0$), onde n é o número de vertices e m o número de arestas. Nas m linhas seguintes contém dois inteiros a e b ($1 \leqslant a, b \leq n, a \neq b$) indicando que os vertices a e b são conectados por uma aresta não direcionada. Não existe mais de uma aresta conectando dois vertices.

Seu programa deve ecerrar a execução quando n e m forem iguais à 0 (zero).

Saída

A cada caso de teste imprimir separados por linha a quantidade de ciclos.

Exemplo:

Entrada	Saída
4 6 1 2 1 3 1 4 2 3 2 4 3 4 0 0	7