Caderno de Provas – *Queijo Francês*

2^a Seletiva Interna – 2018/2

UDESC–Joinville e IFMS–Aquidauana

Servidor BOCA (Arena Joinville):
http://200.19.107.67/boca/

Organização e Realização:

Claudio Cesar de Sá (coordenação geral), Peter Laureano Brendel (coordenação técnica), Daniela Marioti, Gabriel Hermann Negri, Lucas Hermann Negri, Rogério Eduardo da Silva, Diego Buchinger, Roberto Rosso.

Patrocinador 2018: Linx

Lembretes:

- Aos *javaneiros*: **o nome da classe deve ser o mesmo nome do arquivo a ser submetido**. Ex: **classe petrus**, nome do arquivo **petrus.java**;
- Exemplo de leitura de entradas que funcionam:

```
Java: (import java.util.Scanner)
Scanner in = new Scanner(System.in);
ou
Scanner stdin = new Scanner(new BufferedReader(new InputStreamReader(System.in)));

C: (#include <stdio.h>)
int integer1; scanf("%d", &integer1);

C++: (#include <iostream>)
int integer1; std::cin >> integer1;
```

  Exemplo de saída de entradas:

```
Java: System.out.format("%d %d\n", integer1, integer2);
C: printf("%d %d\n", integer1, integer2);
C++: std::cout << integer1 << " " << integer2 << std::endl;
```

- É permitido consultar livros, anotações ou qualquer outro material impresso durante a prova;
- A correção é automatizada, portanto, **siga atentamente as exigências da tarefa quanto ao formato da entrada e saída conforme as amostras dos exemplos**. Deve-se considerar entradas e saídas padrão;
- Todos os compiladores (Java, Python, C e C++) são padrões da distribuição Ubuntu versão 16.04 (gcc C11 habilitado);
- Procure resolver o problema de maneira eficiente. Se o tempo superar o limite pré-definido, a solução não é aceita. As soluções são testadas com outras entradas além das apresentadas como exemplo dos problemas;
- Teste seu programa antes de submetê-lo. A cada problema detectado (erro de compilação, erro em tempo de execução, solução incorreta, formatação imprecisa, tempo excedido ...), há penalização de 20 minutos. O tempo é critério de desempate entre duas ou mais equipes com a mesma quantidade de problemas resolvidos;
- Utilize o *clarification* para dúvidas da prova. Os juízes podem **opcionalmente** atendê-lo com respostas acessíveis a todos;
- Algumas interfaces estão disponíveis nas máquinas Linux, que podem ser utilizada no lugar da *Unity*. Para isto, basta dar *logout*, e selecionar a interface desejada. Usuário e senha: *udesc*;
- Ao pessoal local: **cuidado com os pés sobre as mesas para não desligarem nenhum estabilizador/computador de outras equipes**!
- Na sequência do *Pão de Queijo*, segue a *Queijo Francês*, é uma homenagem a França vencedora da Copa do Mundo de Futebol;

Patrocinador e Agradecimentos

- Linx – Patrocinador oficial do ano de 2018;
- DCC/UDESC;
- Aos bolsistas deste ano pelo empenho;
- Alguns, muitos outros anônimos.

Queijo Francês

2^a Seletiva Interna da UDESC e IFMS–Aquidauana – 2018

18 de agosto de 2018

Conteúdo

Atenção quanto aos nomes e números dos problemas!!!

1 Problema A: Ataque dos Cavalos

`Arquivo: cavalos.[c|cpp|java|py]` `Tempo limite: 10 s`

O prof Kariston do DCC é um amante do xadrez e tem seu reconhecido programa de extensão aqui na UDESC: o **NEXT**. Ele ama tanto o xadrez que decidiu decorar sua casa com peças de xadrez. A sua sala aqui no DCC possui impecáveis arranjos de peças de xadrez e correlatos. Até aqui tudo é verdade!

Contudo, na sua casa, há um enorme corredor (tipo o da sala F209) a ser decorado; ele decidiu usar as peças de cavalos do jogo de xadrez. Na sua casa, este corredor é revestido por belos quadrados de cerâmica (azulejos) de `M` linhas por `N` colunas. Assim, ele quer colocar alguns cavaleiros em algumas (ou possivelmente nenhuma) dessas células. Cada célula conterá no máximo um cavaleiro ou nenhum.

O detalhe especial em seu arranjo é que nenhum par de cavaleiros pode atacar um ao outro. Lembrando que dois cavaleiros ou cavalos podem atacar um ao outro, se forem colocados em células de cantos opostos em formato de um retângulo de 2×3.

Problema

Dada a dimensão do longo corredor ($M \times N$), sua tarefa é calcular quantas maneiras o prof Kariston pode organizar suas peças de cavalos. Dois arranjos são considerados diferentes se houver uma célula que contém um cavalo no primeiro arranjo, mas não no outro arranjo.

Entrada

A primeira linha da entrada é o número de casos de teste T ($1 < T \leq 20$). Nas linhas seguintes seguem os T casos de teste. Cada caso de teste consiste em 2 números: M, o número de linhas e N, o número de colunas, respeitando os limites de: $1 \leq M \leq 3$, $1 \leq N \leq 10000$.

Saída

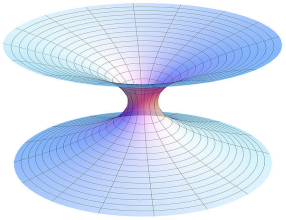
Para cada caso de teste, imprima o número de arranjos possíveis em módulo `1000000009` ($10^9 + 9$).

Exemplo de Entrada	**Exemplo de Saída**
9 1 2 1 1 1 4 2 2 3 1 2 999 3 1000 2 1000 3 876	4 2 16 16 8 829469398 32855896 516094545 144928264

2 Problema B: Buracos Galáticos

`Arquivo: buracos.[c|cpp|java|py]` `Tempo limite: 5 s`

Nosso tempo na terra está findando. Cooper e Amelia se voluntariaram para uma das missões mais importantes da história da humanidade: uma jornada nas estrelas para descobrir novas casas para nossa sociedade.



Felizmente, astrônomos identificaram vários planetas potencialmente habitáveis. Descobriram também que alguns destes planetas possuem buracos de minhoca unindo-os, fazendo com que a distância entre alguns planetas seja efetivamente zero. Para todos os outros planetas, utiliza-se a distância Euclidiana para determinar distâncias.

Problema

Dada a localização da Terra, dos demais planetas e dos buracos de minhoca, encontre o menor caminho de viagem entre quaisquer pares de planetas.

Entrada

- A primeira linha contém um inteiro T ($1 \leq T \leq 10$), representando o número de casos de teste.
- Cada caso consiste da descrição dos planetas, buracos de minhoca e um conjunto de perguntas sobre a distância de viagem.
- Cada caso de teste inicia com a lista de planetas, contendo um inteiro p ($1 \leq p \leq 60$), o número de planetas. Na sequência existem p linhas, cada uma contém o nome de um planeta junto com as suas coordenadas x, y e z, números inteiros ($0 \leq x, y, z \leq 2 \times 10^6$), em *parsecs*. O nome dos planetas consiste de letras e números ASCII, iniciando com uma letra ASCII. O nome dos planetas é sensível a maiúsculas e minúsculas, e contém até 50 caracteres.
- A lista de buracos de minhoca inicia com um inteiro w ($0 \leq w \leq 40$), o número de buracos, seguida por w linhas. Cada linha contém dois nomes de planetas separados por um espaço, indicando uma conexão de mão única entre eles (entrada pelo primeiro planeta e saída pelo segundo).
- A lista de perguntas inicia com o inteiro q ($1 \leq q \leq 20$), o número de perguntas. Cada linha seguinte contém uma pergunta, composta por dois nomes de planetas (existentes) separados por um espaço.

Saída

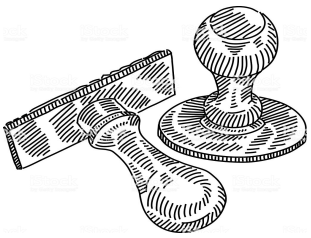
Para cada caso, imprima uma linha contendo *Caso i:*, o número do i-ésimo caso. Então, para cada pergunta no caso em questão, imprima uma linha que diz *A distancia de planeta1 para planeta2 eh d parsecs.”*, substituindo *planeta1* e *planeta2* pelos nomes dos planetas em questão, e d é a menor distância de viagem entre eles, arredondada para o inteiro mais próximo.

Exemplo de Entrada	**Exemplo de Saída**
3 4 Terra 0 0 0 Proxima 5 0 0 Barnards 5 5 0 Sirius 0 5 0 2 Terra Barnards Barnards Sirius 6 Terra Proxima Terra Barnards Terra Sirius Proxima Terra Barnards Terra Sirius Terra 3 z1 0 0 0 z2 10 10 10 z3 10 0 0 1 z1 z2 3 z2 z1 z1 z2 z1 z3 2 Marte 12345 98765 87654 Jupiter 45678 65432 11111 0 1 Marte Jupiter	Caso 1: A distancia de Terra para Proxima eh 5 parsecs. A distancia de Terra para Barnards eh 0 parsecs. A distancia de Terra para Sirius eh 0 parsecs. A distancia de Proxima para Terra eh 5 parsecs. A distancia de Barnards para Terra eh 5 parsecs. A distancia de Sirius para Terra eh 5 parsecs. Caso 2: A distancia de z2 para z1 eh 17 parsecs. A distancia de z1 para z2 eh 0 parsecs. A distancia de z1 para z3 eh 10 parsecs. Caso 3: A distancia de Marte para Jupiter eh 89894 parsecs.

3 Problema C: Carimbos

`Arquivo: carimbos.[c|cpp|java|py]` `Tempo limite: 5 s`

Burocratas adoram burocracia, e isto se traduz em uma tremenda quantidade de papelada! Quando a papelada está completa, o burocrata carimba o documento oficial com o carimbo oficial do escritório. Alguns burocratas são minuciosos, e carimbam o documento múltiplas vezes. Estamos interessamos nos burocratas que carimbam os documentos duas vezes. O carimbo de burocrata utiliza uma determinada área retangular.



Vejamos abaixo um exemplo de carimbo de um burocrata:

```
..#..#..
.######.
..#..#..
```

O símbolo '#' no carimbo cobre uma célula do papel com tinta preta, já o '.' não deixa nenhuma marca (não altera) na célula correspondente do papel. Quando um documento é carimbado pela segunda vez, este novo carimbo pode ser movido para uma outra posição, mas nunca é rotacionado, isto é, o segundo carimbo sempre ficará alinhado nos eixos. Pintar uma célula do papel duas vezes resulta no mesmo que pintar uma célula uma única vez. Ou seja, é indistinguível uma tinta preta, '#', sobre um outro '#'.

Problema

Você receberá um papel que foi carimbado duas vezes. Sua tarefa é determina o menor número de '#'s que poderiam existir no carimbo utilizado originalmente. Garante-se que o papel foi carimbado exatamente duas vezes, e que o carimbo sempre esteve completamente contido no documento.

Por exemplo, o carimbo ilustrado acima, poderia ter deixado a seguinte marca no papel, após ser aplicado duas vezes:

```
..#..#..
.######.
.######.
..#..#..
```

Entrada

Um número T é fornecido, representando o número de documentos para analisar ($1 \leq T \leq 100$). Para cada documento, existirão dois número L e W representando o comprimento e a largura do papel ($1 \leq W,\ L \leq 300$). As próximas L linhas conterão cada uma palavra de comprimento W. Esta palavra é formada pelos caracteres '.' (ausência de tinta) e '#' (presença de tinta).

Saída

Para cada documento, imprima o menor número de '#' possível utilizados sobre o carimbo original.

Exemplo de Entrada	**Exemplo de Saída**
5 4 8 ..#..#.. .######. .######. ..#..#.. 3 3 ... .#. ... 2 6 .##### #####. 2 5 .#.#. #.#.# 6 6 ###.## #.#### ###### ###### #.#### ######	8 1 5 3 21

PS: o nosso burocrata é lento em seus carimbos, mas, seu algoritmo deve executar numa complexidade inferior a $O(n^3)$. Ou seja, *entenda o algoritmo* do burocrata para passar o problema!

4 Problema D: Diamantes

`Arquivo: diamantes.[c|cpp|java|py]` `Tempo limite: 5 s`

O valor total de um diamantes é determinado por sua massa (em UD - *unidades diamânticas*) e também por seu índice de imperfeição. Assim, um diamante grande com várias imperfeições não vale tanto quanto um menor que não possua falhas.

O índice de imperfeição de um diamante pode ser descrito numa escala de 0,0 a 10,0, adotada pela ACME (Associação Corporativa de Minérios Especiais), na qual o valor 0,0 representa um diamante perfeito e 10,0 representa um diamante no limite de imperfeições (pior qualidade).

A empresa *Lucy in the Sky* (de honestidade duvidosa), especializada em tráf..., digo, negociações de diamantes, solicita que um grupo de maratonistas desenvolva um programa que a ajude a organizar seus dados e identificar sequências de diamantes interessantes para os compradores. Como pagamento, a empresa está disposta a pagar ... *nada!*, argumentando que a experiência a ser obtida com o trabalho é a maior recompensa.

Problema

Dada uma sequência de N diamantes, cada uma com peso w_i em UD, e índice de imperfeição c_i na escala de 0,0 a 10,0, encontre a maior subsequência de diamantes para a qual o peso e o índice de imperfeição se tornam, ambos, estritamente mais favoráveis ao comprador. Por exemplo, observe a seguinte sequência de diamantes:

i	w_i	c_i
1	1,5	9,0
2	2,0	2,0
3	2,5	6,0
4	3,0	5,0
5	4,0	2,0
6	10,0	5,5
7	8,0	4,5

Neste caso, a maior subsequência desejada é

i	w_i	c_i
1	1,5	9,0
3	2,5	6,0
4	3,0	5,0
5	4,0	2,0

pois os pesos são estritamente crescentes enquanto os índices de imperfeição são estritamente decrescentes.

Entrada

A entrada inicia com uma linha contendo um único inteiro T , $1 \leq T \leq 100$, indicando o número de casos de teste. Cada caso de teste inicia com uma linha contendo um único inteiro N , $1 \leq N \leq 200$, indicando o número de diamantes. A seguir, para cada caso de teste, são dadas N linhas com 2 número reais cada, w_i e c_i , $0 \leq w_i, c_i \leq 10,0$, indicando a massa em UD e o índice de imperfeição do diamante i. Observe que o diamante pode possuir massa útil desprezível, isto é, w_i pode ser 0.

Saída

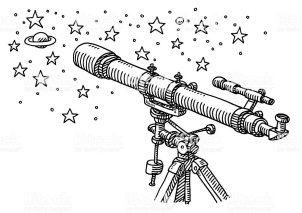
Para cada caso de teste, imprima uma linha com um número inteiro, indicando o tamanho da maior subsequência encontrada em que os pesos são estritamente crescentes enquanto os índices de imperfeição são estritamente decrescentes.

Exemplo de Entrada	**Exemplo de Saída**
4 2 1.0 1.0 1.5 0.0 3 1.0 1.0 1.0 1.0 1.0 1.0 6 1.5 9.0 2.0 2.0 2.5 6.0 3.0 5.0 4.0 2.0 10.0 5.5 3 1.5 1.9 2.0 2.0 1.0 2.1	2 1 4 1

5 Problema E: Contando Estrelas

`Arquivo: estrelas.[c|cpp|java|py]` `Tempo limite: 2 s`

Um eclipse acontece quando o Sol, a Terra e a Lua se alinham. Isso faz com que a Terra fique diretamente entre o Sol e a Lua, bloqueando a luz solar. O eclipse acontece porque a Lua entra na sombra criada pela Terra. Um eclipse famoso foi o do dia 27/07/2018 e teve a duração de aproximadamente 100 minutos.



O diretor Fragalli, entusiasta em ficar vendo o céu com telescópios aqui do CCT, não ia perder esta chance. Contudo, Joinville chove, ora fica nublado devido a serra, e naquele dia, no horário do eclipse, o eclipse frustou a torcida ali organizada.

Assim, o professor Fragalli, teve uma outra ideia junto aos alunos ali frustrados em ver o eclipse. Ele bolou uma brincadeira: **contar as estrelas!** E assim ele definiu áreas no céu, e os alunos deveriam contar quantas estrelas estavam vendo naquelas áreas.

Problema

Considere o céu sobre um espaço cartesiano. Neste espaço há os limites de um retângulo definido por retas paralelas aos eixos x y. Ufa, uma simplificação do problema. O objetivo é contar quantas estrelas estão neste retângulo, incluindo as estrelas que se encontrarem neste limite retangular. Outra simplificação: o campo visual deste telescópio é retangular!

Entrada

A entrada inicia com uma linha com um inteiro C, $1 \leq C \leq 10$, indicando o número de céus (áreas retangulares) a serem avaliados.

Na linha seguinte, segue a definição dos C retângulos ou céus, onde cada retângulo é definido por 4 valores inteiros, nesta ordem: x_{inf} (reta à esquerda), x_{sup} (reta à direita), y_{inf} (reta inferior), y_{sup} (reta superior).

Em seguida a definição dos retângulos, seguem as coordenadas x e y das estrelas existentes em todo céu naquele momento. Estas entradas se encerram com `0 0`. Os limites deste céu, coordenadas das estrelas, e limites dos retângulos, são valores inteiros simples.

Saída

Imprima o número de escrelas em cada um dos C céus no formato:
`CEU #i:  k estrelas`
Acompanhe os exemplos abaixo.

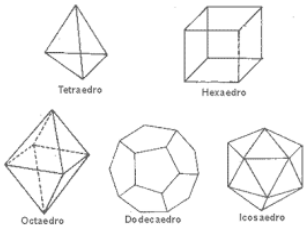
Exemplo de Entrada	**Exemplo de Saída**
2 3 5 2 8 -8 0 -2 2 5 2 4 3 3 2 -3 -2 0 0	CEU #1: 3 estrelas CEU #2: 1 estrelas

6 Problema F: Faces do Poliedro

`Arquivo: faces.[c|cpp|java|py]` `Tempo limite: 2 s`

Você sabia que, a partir de uma esfera, é possível obter diferentes poliedros convexos, a partir de cortes em sua superfície? Isso não é incrível? Euler também se interessou por isso e assim, definiu uma fórmula que relaciona o número de vértices, V, o número de arestas, E, e o número de faces, F, de um poliedro convexo, como:

$$V - E + F = 2$$



Problema

Dizem que um dos últimos desejos de Euler, em seu leito de morte, foi de que, no futuro, alguém obtivesse um método automatizado de computar o número de faces de um poliedro convexo, a partir do número de vértices e arestas, utilizando sua fórmula. Ou não. De qualquer forma, para celebrar as contribuições matemáticas de Euler, faça um programa que calcule o número de faces de um poliedro convexo utilizando a fórmula de Euler.

Entrada

A entrada é iniciada com uma linha contendo um único inteiro T, indicando o número de casos de testes. Cada caso de teste consiste em uma única linha contendo dois inteiros, V e E ($4 \leq V, E \leq 100$), representando o número de vértices e o número de arestas, respectivamente, de um poliedro convexo, separados por um espaço.

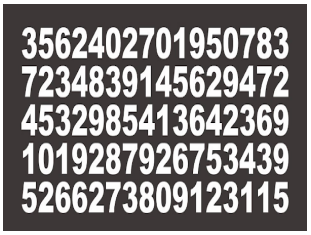
Saída

Para cada caso de teste, imprima uma linha com um número inteiro, indicando o número de faces para o poliedro descrito.

Exemplo de Entrada	**Exemplo de Saída**
2 8 12 4 6	6 4

7 Problema G: Jogo com Números

`Arquivo: jogo.[c|cpp|java|py]` `Tempo limite: 5 s`



O livro Alice no País das Maravilhas (*Alice in Wonderland*) é a obra infantil mais conhecida de Charles Lutwidge Dodgson, um professor de matemática, com contribuições na lógica. Seu pseudônimo o deixou muito famoso: *Lewis Carroll*. Alice foi uma personagem real que faleceu no século passado. Alice era filha de um colega de Charles, de seu trabalho. Alice era criança e tinha um colega de sua idade, o Roberto Rossolino. Qualquer casualidade com o nome do prof aqui do DCC–UDESC, é mera coincidência.

Estes gostavam de brincar e Charles Lutwidge Dodgson criou um jogo para eles. Alice e Roberto estão jogando um jogo em uma linha de `N` quadrados. A linha é inicialmente preenchida com um de cada número de `1` até `N`. Alice e Roberto se revezam na remoção de um único número da linha, sujeitos à restrição de que um número só pode ser removido se não for vizinho de um número maior em qualquer lado. Quando o número é removido, o quadrado que o continha agora está vazio. O vencedor é o jogador que remove o 1 da linha.

Problema

Dada uma configuração inicial, descubra quem vai ganhar, supondo que Alice jogue primeiro e ambos joguem de forma otimizada?

Entrada

A entrada inicia com uma linha com apenas um inteiro T, $1 \leq T \leq 100$, indicando o número de casos de teste. Cada caso de teste começa com uma linha com um único inteiro N, $1 \leq N \leq 100$, indicando o tamanho da linha. Em seguida vem uma linha com os números de 1 a N, separados por espaço, tendo os números organizados da esquerda para a direita.

Saída

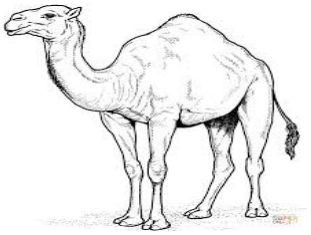
Para cada caso de teste, imprima o nome do jogador(a) vencedor em uma linha.

Exemplo de Entrada	**Exemplo de Saída**
4 4 2 1 3 4 4 1 3 2 4 3 1 3 2 6 2 5 1 6 4 3	Roberto Alice Roberto Alice

8 Problema H: Número Corcova

`Arquivo: corcova.[c|cpp|java|py]` `Tempo limite: 5 s`

Um número do tipo *corcova* (corcova é a lombada que um camelo tem em suas costas) é um número cujos dígitos, possivelmente, sobem (de valor) e então, possivelmente, descem. Porém, nunca o valor desce para depois subir.



- 12321 é um número corcova.
- 101 não é um número corcova.
- 111100001111 não é um número corcova.

Problema

Dado um inteiro `n`, se ele for um *número corcova*, imprima o número de números do tipo corcova menores que `n`. Se ele não for um número corcova, então imprima `-1`.

Entrada

A entrada inicia com uma linha com apenas um inteiro T, $1 \leq T \leq 100$, indicando o número de casos de teste. Em seguida, tem-se os casos de teste composto por um inteiro positivo, maior ou igual ao número `10`. Cada caso de teste será um inteiro positivo de `2` a `70` dígitos escritos em apenas uma linha. O resultado será um inteiro até 64-bits.

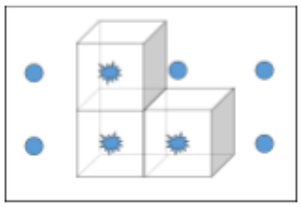
Saída

Para cada caso de teste, imprima `-1` se a entrada não for um número corcova. Imprima o número de números tipo corcova menores que o valor de entrada se o valor de entrada for um número corcova.

Exemplo de Entrada	**Exemplo de Saída**
9 55 101 1000 1234321 10 11 12 333333 2018	55 -1 715 94708 10 11 12 45227 -1

9 Problema I: Queda

`Arquivo: queda.[c|cpp|java|py]` `Tempo limite: 30 s`



Você é um engenheiro assistente que trabalha na manutenção da nave estelar ENTERPRAISSE. Sua função é garantir o bom funcionamento dos motores de dobra espacial da nave. Um motor de dobra consiste basicamente em uma série de reatores de anti-matéria ligados uns aos outros na forma de um grid tridimensional. O problema é que a anti-matéria é uma substância altamente instável e frequentemente ocorrem quedas de energia no sistema de isolamento de alguns dos reatores. Uma queda de energia no isolamento de um reator de anti-matéria é algo preocupante pois pode causar uma explosão se a anti-matéria vazar e entrar em contato com algum reator vizinho ainda em pleno funcionamento. A fim de evitar isso a solução é instalar painéis de isolamento de anti-matéria entre um reator com defeito e seus vizinhos funcionais. Instalar um painel desses é uma operação delicada pois estes precisam estar alinhados com os eixos ortogonais da nave; cada painel leva exatamente 1 minuto para ser instalado. Obviamente, devido ao risco iminente de explosão, o que se quer é que a operação de isolamento seja feita o mais rapidamente possível, mesmo que pra isso seja necessário isolar alguns reatores de anti-matéria sem defeito. A fim de se produzir um isolamento perfeito, um poliedro fechado deve ser construído.

Problema

Dadas as coordenadas das células de anti-matéria com defeito, o capitão espera de você um relatório prevendo quanto tempo demorará o conserto da nave.

Entrada

A entrada começa com um número inteiro $N(1 \leq N \leq 100)$ que indica a quantidade de casos de teste. Cada caso de teste consiste de um valor $F(1 \leq F \leq 100)$ que representa a quantidade de células defeituosas no reator de anti-matéria. Em seguida, são fornecidas F linhas contendo três valores cada: x_i, y_i, z_i cada um no intervalo fechado $[0, 9]$. Cada tripla (x_i, y_i, z_i) é única e representa as coordenadas de uma célula defeituosa que precisa ser isolada.

Saída

Para cada caso de teste, imprima em uma única linha a quantidade mínima de minutos que cada conserto deverá demorar.

Exemplo de Entrada	**Exemplo de Saída**
3 1 0 0 0 2 0 0 0 0 0 1 3 0 0 0 0 0 1 0 1 1	6 10 14

10 Problema J: Runas

`Arquivo: runas.[c|cpp|java|py]` `Tempo limite: 5 s`

Ajude a renomada arqueóloga Luciana Jones a decifrar um conjunto de runas medievais. Ela já descobriu que esta sociedade antiga usava um sistema decimal e que eles nunca começavam um número com zero. Ela decifrou a maioria dos dígitos, bem como alguns dos operadores, porém ela precisa da sua ajuda para concluir o trabalho.

Professora Luciana lhe dará uma simples equação matemática. Ela codificou as runas já conhecidas em números. Os únicos operadores já decifrados são a soma (`+`), a subtração (`-`) e a multiplicação (`*`), portanto esses serão os únicos que aparecerão nas expressões. Cada número é representado no intervalo -999.999 a 999.999 e consistem apenas dos dígitos '0' a '9', possivelmente podendo iniciar por um caracter '-' e conter alguns simbolos '?'. O símbolo '?' representa um dígito da runa o qual a professora não conhece (nunca um operador nem um '=' ou caracter inicial '-'). Todos os '?' representarão o mesmo dígito (`0 - 9`) e não será algum daqueles que já constam da expressão.

Problema

Dada uma expressão, descubra o valor da runa representado pelo símbolo de interrogação. Se mais de um dígito for possível, escolha sempre o menor valor. Se nenhum dígito resolver a expressão, bem, isto é uma má notícia para a professora – significa que ela decifrou algumas runas com erro. Imprima `-1` neste caso.

Entrada

A entrada começa com o número de casos de teste $T(1 \leq T \leq 100)$. Cada caso de teste consistirá de uma única linha na forma:

$$[numero][op][numero] = [numero]$$

Cada [*numero*] consiste de uma sequência de dígito (`0-9`), podendo ou não ser precedido pelo sinal '-', e possivelmente alguns '?'. Nenhum número começará com zero (com exceção do próprio 0), não existe a sequência `-0`, e ainda nenhum número conterá mais dos que 6 caracteres (dígitos ou '?'). O caracter para [*op*] separa o primeiro número do segundo, e poderá assumir os seguintes valores: '+', '-' ou '*'. O símbolo '=' sempre estará presente entre o segundo e o terceiro números. Nenhuma expressão contém espaços em branco, tabulações ou outros caracteres. É garantido que cada expressão conterá pelo menos um símbolo '?'.

Saída

Imprima o menor dígito que resolva a equação quando substituído pelo '?', ou imprima `-1` se nenhum dígito funcionar. Não imprima qualquer outro caracter adicional (como espaços ou linhas em branco).

Exemplo de Entrada	**Exemplo de Saída**
5 1+1=? 123*45?=5?088 -5?*-1=5? 19--45=5? ??*??=302?	2 6 0 -1 5