Jogos de Cravos

1ª Seletiva Interna – 2014/1

Servidor BOCA:

`http://10.20.107.207/boca/` (acesso interno)

`http://200.19.107.207/boca/` (acesso externo)

Organização e Realização:

Claudio Cesar de Sá (coordenação geral), Lucas Hermann Negri (coordenação técnica), Bruno Ribas (UFPr), Ricardo Oliveira (Dinf/UFPR), Yuri Kaszubowski Lopes, Alexandre Gonçalves Silva (revisão técnica), Roberto Silvio Ubertino Rosso Jr., Rogério Eduardo da Silva

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**;
- 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 de seu programa. Deve-se considerar entradas e saídas padrão;
- 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;
- A interface KDE está disponível nas máquinas Linux, que pode ser utilizada ao invés da Unity. Para isto, basta dar *logout*, e selecionar a interface KDE. Usuário e senha: *udesc*;

Patrocinador e Agradecimentos

- Linx – Patrocinador oficial do ano de 2014;
- DCC/UDESC;
- Aos bolsistas deste ano pelo empenho ;
- Alguns, muitos outros anônimos.

Jogos de Cravos

1ª Seletiva Interna da UDESC

25 de abril de 2014

Conteúdo

1 Problema A: Apostas

`Arquivo: apostas.[c|cpp|java]`

Amadeus está construindo um site para gerenciar apostas. A interface é linda, construída com as várias letrinhas que compõem o mundo do desenvolvimento web: CSS9, HTML8, além, é claro, da integração com as diversas redes sociais. A primeira funcionalidade do site será contabilizar o valor total de um bolo de apostas. Porém, Amadeus não faz ideia de como implementar isto! Sua tarefa é ajudar o desavisado Amadeus a implementar sua primeira funcionalidade útil.

Entrada

A entrada é composta por vários casos de teste. Cada caso de teste é representado por uma linha composta por uma sequência de números inteiros (de 1 até 1.000 valores), onde cada número é o valor de uma aposta (cada aposta é um valor inteiro de 1 até 1.000).

A entrada termina no final do arquivo (EOF) (Amadeus não se deu ao trabalho de marcar o número de casos e o número de valores em cada aposta).

Saída

Para cada caso de teste, imprima uma linha contendo o somatório das apostas do caso.

Exemplo de Entrada

```
5
3 10
1 1 1
9 2 4 4 5 6 7
```

Exemplo de Saída

```
5
13
3
37
```

2 Problema B: Banindo Episódios

`Arquivo: banindo.[c|cpp|java]`

Em 16 de Dezembro de 1997, ia ao ar, no Japão, o 38^o episódio de *Pokémon*. Neste episódio, intitulado *Porygon, o guerreiro virtual*, Ash e seus companheiros são transportados para o cyberespaço, onde enfrentam a Equipe *Rocket* com a ajuda de *Porygon*.

Em determinado momento deste episódio, uma dada sequência de cores foi exibida e vários telespectadores, ao assistirem esta sequência de cores particular, acabaram sofrendo ataques e tiveram de ser encaminhados imediatamente aos hospitais japoneses. Devido a este trágico acidente, o episódio foi banido, e nunca mais foi exibido novamente.

Novos episódios da série estão sendo produzidos. Antes de serem exibidos, entretanto, é necessário verificar se a sequência de cores que causou o acidente não está presente nos episódios novos, pois, se estiver, eles também devem ser banidos. Sua tarefa é, dada a sequência de cores problemática e a sequência de cores de um episódio, verificar se a sequência de cores problemática está presente no episódio. Note que, para tal, a sequência de cores problemática deve ser uma *subsequência contínua* da sequência do episódio.

Neste problema, uma cor é representada por uma tripla (r, g, b), com $0 \leq r, g, b < 256$. Estes números indicam, em uma escala de 0 a 255, o tom de vermelho, verde e azul na cor. Assim, a cor vermelha é representada por $(255, 0, 0)$ e a amarela por $(255, 255, 0)$, e a sequência Vermelha,Amarela,Vermelha é representada por $((255, 0, 0), (255, 255, 0), (255, 0, 0))$.

Entrada

Cada caso de teste inicia com uma linha contendo dois inteiros M e N ($1 \leq M \leq N \leq 3 \times 10^5$), onde M é o tamanho da sequência problemática e N é o tamanho da sequência do episódio.

A próxima linha contém $3 \times M$ inteiros entre 0 e 255, inclusive. Os três primeiros inteiros representam a primeira cor da sequência problemática. Os três próximos inteiros representam a segunda cor, e assim por diante. A última linha contém $3 \times N$ inteiros, representando a sequência de cores do episódio de forma análoga.

O último caso de teste é seguido por uma linha contendo dois zeros.

Saída

Para cada caso de teste, imprima *banido* se a sequência problemática ocorre no episódio, e *permitido* caso contrário.

Exemplo de Entrada

```
3 5
0 0 0 255 255 0 255 0 255
255 255 255 0 0 0 255 255 0 255 0 255 255 127 0
3 5
0 0 0 255 255 0 255 0 255
255 255 255 0 0 0 255 255 0 255 127 255 255 127 0
0 0
```

Exemplo de Saída

```
banido
permitido
```

3 Problema C: Calculando

`Arquivo: calculando.[c|cpp|java]`

Agar trabalha como caixa na cantina universitária e, para passar o tempo, decidiu calcular o número de maneiras no qual ele consegue formar um determinado troco. Ele logo percebeu que o número de combinações de valores era grande demais para calcular no guardanapo, e decidiu então fazer um programa para resolver o problema. Porém, Agar está inseguro com sua solução e pediu para você também resolver o problema, podendo assim comparar os resultados.

Por exemplo, se Agar possui os valores 1, 2 e 5 disponíveis, ele pode formar o troco 5 de quatro formas diferentes: $\{1, 1, 1, 1, 1\}$, $\{2, 1, 1, 1\}$, $\{2, 2, 1\}$ e $\{5\}$ (a posição dos valores não importa).

Note que, dependendo dos valores disponíveis para Agar, pode não ser possível compor o troco! Neste caso, o número de combinações é igual a 0, naturalmente.

Entrada

A entrada é composta por múltiplos casos de teste. A entrada é iniciada por uma linha contendo o número de casos de teste T ($1 \leq T \leq 10$).

Cada caso de teste é iniciado por uma linha contendo o valor M ($1 \leq M \leq 20$) que representa o número de valores disponíveis para compor o troco (um mesmo valor pode ser utilizado repetidas vezes). A próxima linha contém M inteiros distintos (entre 1 e 100), representando cada valor disponível. A linha seguinte contém o número de trocos a serem compostos N ($1 \leq N \leq 1.000$), sendo que cada uma das próximas N linhas contém um valor inteiro Q ($1 \leq Q \leq 100.000$) correspondendo ao valor total que deve ser composto.

Saída

Para cada troco, seu programa deve imprimir uma linha contendo o número de combinações possíveis (sendo que a ordem dos valores não importa) para compor o troco Q. Como o número de combinações pode ser muito grande, sempre imprima o resto da divisão inteira do número de combinações por 1003.

Exemplo de Entrada

```
2
3
1 2 5
2
3
5
1
10
2
7
10
```

Exemplo de Saída

```
2
4
0
1
```

4 Problema D: Desfile dos Patos

`Arquivo: desfile.[c|cpp|java]`

Em uma pacata cidade do interior um curioso desfile acontece toda manhã às seis horas. O desfile dos Patos acontece na avenida mais badalada da cidade (a Av. Tupi). Esse desfile é tão reconhecido que pelo menos uma vez por semana a televisão local filma o evento e transmite para a micro-região. Os patos sempre saem para seu desfile alimentados e percorrem a avenida como verdadeiros reis da cidade. Não é por acaso que a cidade possui o Trevo do Patinho com a estátua do mais reconhecido Pato que viveu nessa cidade, o famoso Pato Branco de polainas.

Durante o desfile dessa manhã, Bozena, percebeu que vários Patos possuem uma mecha em suas penas. Essas mechas são um filete de alguma cor. Marciano, um aluno de uma escola local, percebeu que uma cor é a majoritária (mais da metade dos patos tem essa cor) no conjunto de todas as cores nas mechas, porém, Patrick (colega de Marciano) não consegue decidir qual é a cor majoritária, algumas cores parecem ter a maioria por pouca diferença e por isso é difícil saber qual é a majoritária. Então Patrick o desafiou a escrever um programa de computador que dada uma sequência das cores que aparecem nos patos durante o desfile diga qual é a cor majoritária.

Entrada

A entrada possui vários casos de teste. A primeira linha de um caso de teste possui um número N ($1 \leq N \leq 5000$) que representa quantos patos foram observados. A segunda linha de um caso de teste possui N inteiros, a_i ($1 \leq a_i \leq 10^6$), separados por um espaço em branco, correspondendo ao código da cor que estava na mecha do Pato. A entrada termina quando $N = 0$.

Saída

Para cada caso de teste imprima, em uma única linha, o código da cor que é a majoritária no desfile.

Exemplo de Entrada

```
5
1 4 1 2 1
13
1 1 1 3 3 2 2 3 3 3 2 3 3
0
```

Exemplo de Saída

```
1
3
```

5 Problema E: Emprego Duro

`Arquivo: emprego.[c|cpp|java]`

O professor Claudiusvirus costuma contar algumas histórias em suas aulas, que embora choquem alguns estudantes, algumas delas são fatos reais. Algo como Nelson Rodrigues já tinha escrito: *A Vida como Ela é...*

Felizmente, embora alguns custem a acreditar, a dureza começa mesmo no dia que você se forma. Deixaste de ter o seu *status* de estudante, e vais em busca do tão sonhado emprego, naquela multinacional que paga muito mais que qualquer outra empresa de TI.

Nesta nova empresa o seu sistema de seleção é duro e bizarro. Todos os dias chegam muitos candidatos na sala do RH (uma ampla sala). Digamos N candidatos, e todos recebem uma ficha de acordo com a ordem de chegada. Ao final do dia, alguns candidatos devolvem as fichas a secretária. Aqueles que não devolverem foram os escolhidos, e já ficam por lá.

O processo de seleção com entrevista etc, é meio chato, contudo, o responsável do RH, se limita em falar: *você está achando tudo isto chato, espere até você conhecer o seu chefe*!

Entrada

O arquivo de entrada contém na primeira linha um valor inteiro M, o qual corresponde ao número de dias de recrutamento (período do processo seletivo). Na linha seguinte, contém dois inteiros N e R, indicando respectivamente o número de fichas distribuídas no dia e o número de candidatos recusados na seleção. Como esperado, os N candidatos do dia são numerados de 1 a N. A terceira linha, contém uma lista de entrada com R inteiros, indicando os candidatos recusados na seleção.

Saída

Para cada dia de seleção, o seu programa deve conter uma única linha, indicando quais foram os candidatos selecionados, na ordem crescente de suas identificações. Deixe um espaço em branco após cada identificador (note que isto significa que deve haver um espaço em branco após o último identificador). Se todos os candidatos foram recusados no teste do dia, apenas imprima o carácter '$*$' (asterisco).

Restrições

- $1 \leq M \leq 30$
- $1 \leq R \leq N \leq 10^4$

Exemplo de Entrada

```
3
5 3
3 1 5
6 6
6 1 3 2 5 4
7 3
5 2 1
```

Exemplo de Saída

```
2 4
*
3 4 6 7
```

6 Problema F: Fim do Desmatamento

`Arquivo: desmata.[c|cpp|java]`

Uma matriz de A linhas e B colunas, resultado do processamento de uma imagem de satélite, é formada por inteiros sem sinal de 3 bits. Ou seja cada elemento, na linha x e coluna y, pode assumir um entre oito valores diferentes, associado à região em que faz parte na imagem:

0 - Cidade
1 - Nuvem
2 - Mata
3 - Rio/Lago/Mar
4 - Neve
5 - Montanha
6 - Deserto
7 - Outra

Um órgão do governo de Santa Catarina pretende, a partir de duas destas matrizes pré-processadas a partir de imagens de uma mesma região de Joinville (uma de 1987 e outra de 2007 com coordenadas geográficas idênticas), verificar todos os pontos de desmatamento (d) e de reflorestamento (r) e, ao final, ter uma índice de impacto ambiental. Tal índice é obtido pela *quantidade de elementos de desmatamento* dividido pela *quantidade de elementos de reflorestamento*, com duas casas decimais. Observe que há, portanto, interesse em verificar a presença ou ausência de elementos da matriz com valor igual ou não a 2 de acordo com a lista de significados descrita anteriormente.

Entrada

A entrada é constituída apenas por números inteiros divididos da seguinte forma: (a) linha inicial com dois valores, sendo o primeiro referente ao número de linhas (A) da matriz e o segundo, o número de colunas (B); (b) linha seguinte com o ano de aquisição da primeira imagem de satélite; (c) A linhas seguintes com os valores da primeira matriz; (d) linha seguinte com o ano de aquisição da segunda imagem de satélite; (e) B linhas seguintes com os valores da segunda matriz.

Tamanho das matrizes: $1 \leq A,\ B\ \leq 300$.

Saída

A saída é construída nesta sequência: (a) primeira linha com dois valores indicando número de linhas (A) e de colunas (B); (b) segunda linha com primeiro ano e segundo ano; (c) terceira linha em diante com uma matriz de caracteres, no qual cada elemento pode ser:

- **.** se não houve alteração
- **r** se houve reflorestamento
- **d** se houve desmatamento

E, por fim, (d) última linha contendo o índice de impacto ambiental pela cardinalidade(r) / cardinalidade(d) com arredondamento em duas casas decimais.

Exemplo de Entrada

```
3 5
1987
7 7 0 0 2
7 3 2 0 0
2 2 3 0 2
2007
7 2 0 0 0
7 3 0 0 0
2 2 3 2 0
```

Exemplo de Saída

```
3 5
1987 2007
.r..d
..d..
...rd
0.67
```

7 Problema G: *Games*

`Arquivo: games.[c|cpp|java]`

O Reino Unido é reconhecido como um dos países de maior influência cultural no mundo. Um típico exemplo são os mais variados esportes criados (ou adaptados) pelos *mates* que habitam a ilha britânica e arredores e que depois são adotados ao redor do planeta. A lista vai desde o popular futebol, passando por criações um tanto quanto estranhas como *cheese rolling* (corrida do queijo) e *shin-kicking* (chuta-canela) até esportes ainda não tão populares fora das "ensolaradas"ilhas da região como críquete e badminton. Na realidade, o famoso pesquisador e historiador Inglês Sir Noth Ingbet Tertodo, argumenta que a interessante relação entre a popularização de esportes britânicos e o decréscimo de conquistas esportivas dos britânicos ainda tem causa desconhecida. Ainda, um importante aspecto da pesquisa, ressalta o professor, é a relação entre a criação de novos esportes e o crescimento das conquistas. O único esporte que a anos insiste em contrariar todas as demais amostragens é o golfe de *pub*[1], o qual os ingleses sustentam a sua hegemonia facilmente mesmo depois da sua popularização.

Um dos esportes que está sofrendo já a algum tempo o efeito da popularização é o Badminton. Neste processo as regras da pontuação do jogo acabam sendo bastante variadas de um país para o outro e a associação internacional de badminton está sendo bastante flexível quanto a isso. O único ponto em comum é que o jogo é disputado em um número de sets, S, que termina quando um jogador alcançar um número de pontos P desde que a diferença de pontos seja igual ou maior a 2 pontos. Se o valor de P for excedido sem que haja uma diferença de 2 pontos o jogo continua até que tal diferença seja obtida. Os valores de S e P podem ser definidos diferentemente para cada país.

A associação mantém um banco de dados com a ordem de pontuação e resultados de todos os jogos disputados. Bem, mantinha, até que algum estagiário acreditou ser bem interessante rodar o comando "*DROP TABLE result_sets*". Sua tarefa é utilizar a ordem de pontuação para reconstruir os placares perdidos.

Entrada

A entrada possui vários casos de teste. A primeira linha de um caso de teste possui um número inteiro P ($3 \leq P \leq 256$) que representa o número de pontos necessários para se vencer um set, quando a diferença para a pontuação do adversário for de pelo menos 2 pontos. A segunda linha de um caso de teste possui os caracteres 'A' ou 'B' indicando qual jogador pontuou na jogada. Cada linha representa um jogo com um número $S \geq 0$ de sets, a entrada está cronologicamente ordenada. A entrada termina quando $P = 0$.

Saída

Para cada caso de teste imprima, em uma única linha, o placar do jogo no formato "*A X B*", sem as aspas (veja exemplo), onde A é o número de sets vencidos por A e B o número de sets vencidos por B. Caso não seja possível reconstruir o placar imprima "corrompido".

[1] Jogado com 9 ou 18 "buracos"(cada um sendo um diferente *pub*) os jogadores devem tomar um pint (unidade de medida equivalente a 568ml de cerveja) em um número pré-determinado de goles (chamado *par*). Cada gole a mais ou a menos é contabilizado e assim como no golfe tradicional aquele com o menor número de pontos é o vencedor

Exemplo de Entrada

```
3
AAABBBABAA
3
AAAABAA
3
ABBBBABB
5
ABABABAABBBBBABABABAA
5
ABABABABABABBBAAAAABBBBB
3
ABABABAAB
0
```

Exemplo de Saída

```
2 X 1
2 X 0
0 X 2
2 X 1
1 X 2
corrompido
```

8 Problema H: Heroica Viagem de Hermes

`Arquivo: heroica.[c|cpp|java]`

Havia um homem chamado Hermes, cujo objetivo na vida era ser um campeão, um herói. Dotado de uma incrível agilidade e resistência física, decidiu atuar como mensageiro do exército. No exército, Hermes sempre escolheu as rotas mais longas, deixando as outras para outros mensageiros.

Porém, o capitão do exército teve um grande problema: ele não sabia identificar qual a rota mais longa para Hermes! Assim, ele requisitou sua ajuda[2] para identificar qual a maior rota entre um conjunto de cidades e estradas. Devido as restrições relacionadas a tarefa de mensageiro, no conjunto de cidades e estradas fornecido pelo general, as estradas são direcionadas e não existem ciclos.

Entrada

A entrada é composta por vários casos de teste. Cada caso de teste é iniciado por dois inteiros N e M ($2 \leq N \leq 1.000$, $1 \leq 30.000$) demarcando o número de cidades e o número de estradas de uma determinada região. As próximas M linhas contém as informações sobre as M estradas, sendo que cada linha contém três inteiros A, B e C, representando que existe uma estrada da cidade A para a cidade B com distância de C (unidade arbitrária). As cidades são representadas por inteiros entre 1 e N, sendo que $1 \leq C \leq 1.000$.

A entrada termina quando $N = M = 0$, sendo que esta entrada não deve ser processada.

Saída

Para cada caso de teste, imprima uma linha contendo o comprimento do maior caminho encontrado. Perceba a cidade de início e de fim da jornada não são fixas.

Exemplo de Entrada

```
4 4
1 2 5
1 3 5
3 2 2
2 4 10
0 0
```

Exemplo de Saída

```
17
```

[2] Não importa como!

9 Problema I: Indefesas Estátuas

`Arquivo: indefesas.[c|cpp|java]`

Felix João Humilde é um *nerd* rico residente em uma cidade pequena do Centro-Oeste. Ele erigiu quatro estátuas em sua homenagem no parque da cidade (que por acaso pertence a ele). Felix é muito orgulhoso das estátuas. Mas agora está preocupado com vândalos, crianças pequenas com chicletes e com os cães com infecções urinárias. Para resolver esse problema, ele decidiu construir uma cerca em volta das estátuas (Felix também é proprietário da empresa local de construção de cercas). Por várias razões estéticas ele gostaria de ter as seguintes condições atendidas:

1. O espaço fechado tem de ser um quadrado;
2. A distância entre cada estátua e seu lado mais próximo da cerca de vedação deve ser de 5 pés;
3. Duas estátuas não devem ter o mesmo lado da cerca como o mais próximo.

Depois de trabalhar por um total de 12 segundos, Felix percebeu que ele não tinha a menor ideia do comprimento de cerca necessário. Nem mesmo se as condições acima podem ser satisfeitas. Desde que ele está planejando homenagens similares em outros parques que possui, ele gostaria que alguém escrevesse um programa para resolver este problema (que mais tarde Felix irá tomar o crédito para si, é claro). Obs.: O pé (*foot*) é uma medida de comprimento do sistema imperial britânico e equivale a 30.48 centímetros.

Entrada

A primeira linha do arquivo de entrada irá conter um inteiro `n` indicando o número de casos de teste. Os casos de teste vem em seguida, um por linha, consistindo cada um de oito valores inteiros que são as coordenadas `x` e `y` da primeira, segunda, terceira e quarta estátua. Todos os valores são em pés e ficarão entre `-100` e `100` (valores inteiros). Duas estátuas não podem estar na mesma localização.

Saída

Para cada caso de teste imprima o número do caso seguido de uma das duas respostas:

1. Se tiver uma solução, imprima o comprimento do lado do quadrado que contém as estátuas. Se existirem múltiplas soluções, imprima o comprimento do lado da solução de maior tamanho.
2. Se não for possível construir uma cerca que cumpra as exigências de Felix, imprima:
   `Caso ...: Sem solucao`
   Todos os resultados numéricos impressos devem ser arredondados para o centésimo de pé mais próximo. Siga os formatos dos exemplos abaixo.

Exemplo de Entrada

```
4
0 1 2 5 5 4 3 0
10 10 30 30 20 20 0 10
0 1 1 0 3 4 4 2
0 1 0 2 0 3 0 4
```

Exemplo de Saída

```
Caso 1: 15.00
Caso 2: Sem solucao
Caso 3: 14.00
Caso 4: Sem solucao
```

10 Problema J: Jogo do Maior

`Arquivo: jogo.[c|cpp|java]`

Ogg gosta muito de brincar com seus filhos. Seu jogo preferido é o *jogo do maior*, de autoria própria. Este passatempo (no tempo das cavernas se tinha muito tempo disponível para jogos) é jogado em dupla, Ogg e um dos seus filhos. O jogo procede da seguinte forma: os dois participantes escolhem um número de rodadas e, a cada rodada, cada participante diz um número de 1 até 9 em voz alta, sendo que o participante que falar o número mais alto ganha um ponto (em caso de empate, ninguém ganha o ponto). No final das rodadas, os pontos são contabilizados e o participante com o maior número de pontos ganha. Ogg e seus filhos gostam muito do jogo, mas se perdem na contagem dos pontos. Você conseguirá ajudar Ogg a verificar a pontuação de uma lista de jogos?

Entrada

A entrada é composta por vários casos de teste (partidas). Cada caso é iniciado com um inteiro N (de 0 até 10) representando o número de rodadas da partida, sendo que o valor 0 representa o final da entrada e não deve ser processado. Cada uma das próximas N linhas contém dois inteiros, A e B, onde A é o número escolhido pelo primeiro jogador e B é o número escolhido pelo segundo jogador ($0 \leq A, B \leq 10$).

Saída

A saída deve ser composta por uma linha por caso de teste, contendo o número de pontos de cada jogador, separados por um espaço.

Exemplo de Entrada

```
3
5 3
8 2
5 6
2
5 5
0 0
0
```

Exemplo de Saída

```
2 1
0 0
```

11 Problema L: Laboratory Tree

`Arquivo: laboratory.[c|cpp|java]`

It will soon be time to decorate the christmas tree at the laboratory. The professors are already debating the optimal way to put decorations in a tree. They agree that it is essential to distribute the decorations evenly over the branches of the tree.

This problem is limited to binary christmas trees. Such trees consist of a trunk, which splits into two subtrees. Each subtree may itself split further into two smaller subtrees and so on. A subtree that does not split any further is a twig. A twig may be decorated by attaching at most one ball to it.

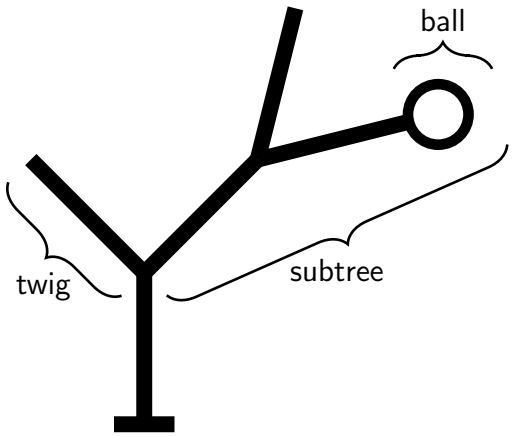

Figura 1: Example of a tree with subtrees, twigs and one ball.

A decorated tree has an even distribution of balls if and only if the following requirement is satisfied:
At every point where a (sub)tree splits into two smaller subtrees t_1 and t_2, the total number of balls in the left subtree $\mathrm{N}(t_1)$ and the total number of balls in the right subtree $\mathrm{N}(t_2)$ must either be equal or differ by one. That is: $\mid \mathrm{N}(t_1) - \mathrm{N}(t_2) \mid \leq 1$.

In their enthusiasm, the professors initially attach balls to arbitrary twigs in the tree. When they can not find any more balls to put in the tree, they stand back and consider the result. In most cases, the distribution of the balls is not quite even. They decide to fix this by moving some of the balls to different twigs.

Task

Given the structure of the tree and the initial locations of the balls, calculate the minimum number of balls that must be moved to achieve an even distribution as defined above.

Note that it is not allowed to add new balls to the tree or to permanently remove balls from the tree. The only way in which the tree may be changed is by moving balls to different twigs.

Input

For each test case, the input consists of one line describing a decorated tree.

The description of a tree consists of a recursive description of its subtrees. A (sub)tree is represented by a string in one of the following forms:

- The string '()' represents a twig without a ball.
- The string '(B)' represents a twig with a ball attached to it.
- The string '(t_1t_2)' represents a (sub)tree that splits into the two smaller subtrees represented by t_1 and t_2, where t_1 and t_2 are strings in one of the forms listed here.

A tree contains at least 2 and at most 1000 twigs.

Output

For each test case, print one line of output.

If it is possible to distribute the balls evenly through the tree, print the minimum number of balls that must be moved to satisfy the requirement of even distribution.

If it is not possible to distribute the balls evenly, print the word *impossible*.

Sample

Input	Output
`((B)())`	`0`
`((((B)(B))((B)()))(B))`	`impossible`
`(()(((B)(B))(B)))`	`1`

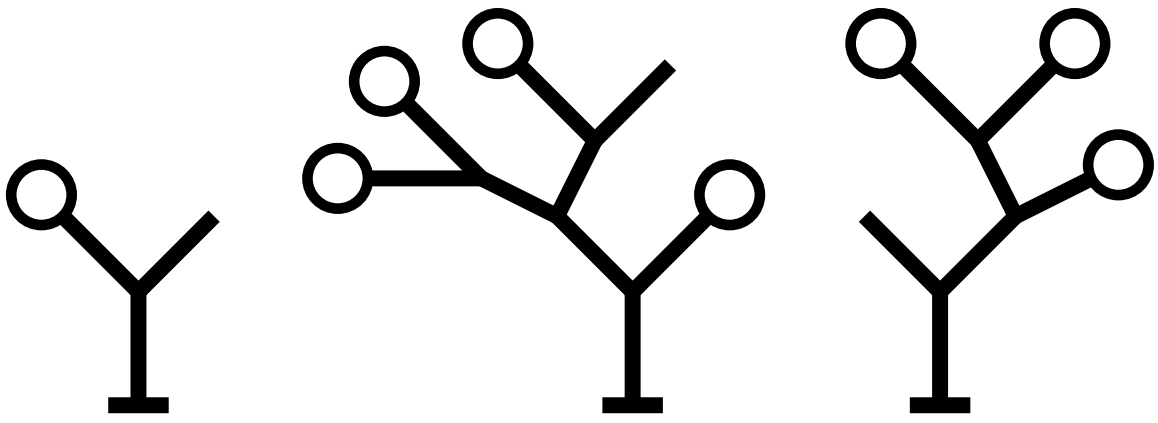

Figura 2: Trees corresponding to the example input cases.