MODALIDADE AVANÇADA CATEGORIAS: JÚNIOR E SÊNIOR

A PROVA TERÁ DURAÇÃO DE **QUATRO HORAS**

Este Caderno contém 7 problemas
16 de Agosto de 2019

Instruções

LEIA ATENTAMENTE AS INSTRUÇÕES ABAIXO ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto de 12 páginas . Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É permitido consultar livros, anotações ou qualquer outro material impresso durante a prova.
- É permitida a consulta ao menu ajuda do ambiente de programação se este estiver disponível.
- A correção é automatizada, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa.
- Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc.), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
- As tarefas não estão ordenadas, neste caderno, por ordem de dificuldade; procure resolver as mais fáceis primeiro.
- Preste muita atenção no nome dos arquivos fonte indicados nas tarefas. Soluções na linguagem C devem ser arquivos com sufixo .c ; soluções na linguagem C++ devem ser arquivos com sufixo .cpp; soluções na linguagem Java devem ser arquivos com sufixo . java e a classe principal deve ter o mesmo nome do arquivo fonte; e soluções na linguagem Python 2.7 devem ser arquivos com sufixo .py.
- Todos os dados devem ser lidos da entrada padrão (normalmente é o teclado) e escritos na saída padrão (normalmente é a tela). Utilize as funções padrão para entradas e saída de dados:
    - em C: scanf, getchar, printf, putchar;
    - em Pascal: readln, read, writeln, write;
    - em C++: as mesmas do C ou os objetos cout e cin;
    - em Java: qualquer classe ou função padrão, como por exemplo Scanner, BufferedReader, BufferedWriter e System.out.println;
    - **em Python 2.7 ou Python 3.6 : read, readline, readlines, print, write, raw_input e input.**

- Procure resolver o problema de maneira eficiente. Na correção, Eficiência também será levada em conta. Em cada problema é indicado o tempo limite. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.
- Para a realização da prova entre no endereço <https://boca.laced.com.br/>, faça seu login e boa sorte!

Problema A. Sequências

Nome do Programa: A.(c | pas | cpp | java | py)
Tempo: 1 segundo

A sequência Fibonnaci é formada por números inteiros, começando por 0 e 1 onde os valores são criados a partir da soma dos dois valores anteriores, como em **0,1**, **1, 2, 3, 5, 8, 13, 21, …** .

A Progressão Aritmética(PA) é uma sequência a qual cada número, a partir do segundo, é formado pela soma do anterior e de uma constante, como em **4, 7, 10, …** .

A Progressão Geométrica(PG) é uma sequência a qual cada número, a partir do segundo, é formado pelo produto do anterior e uma constante, como em **2, 6, 18, …** .

O seu objetivo é identificar o tipo da sequência e qual os próximos dois números da mesma.

Entrada

Cada caso de teste é composto por uma linha contendo quatro números inteiros distintos separados por espaços em branco.

Saída

Para cada caso de teste, indique se é Fibonnaci(imprima "FI"), Progressão Aritmética(imprima "PA") ou Progressão Geométrica(imprima "PG"), seguida de um espaço em branco e os dois próximos números da sequência separados por espaço.

Exemplos de Entrada

Entrada	**Saída**
4 7 10 13	PA 16 19

Entrada	**Saída**
1 2 4 8	PG 16 32

Entrada	**Saída**
2 3 5 8	FI 13 21

Problema B. Roupas

Nome do Programa: B.(c | pas | cpp | java | py)
Tempo: 1 segundo

Railson comprou uma máquina de lavar roupas para uso de sua família que funciona pela quantidade de peças colocadas dentro, podendo ser dividida em 3 níveis: 1- baixo (14 peças ou menos), 2- médio (15 até 28 peças) e 3- alto (29 até 56 peças), preferindo colocar a máquina sempre nos níveis médio ou alto, economizando energia. Nos dias em que lava as roupas, Railson separa as peças em coloridas, pretas e brancas.
Ele pediu a sua ajuda para juntar as roupas dos seus familiares, formando o maior nível possível dependendo da quantidade de peças sujas deles. Seu objetivo é solucionar isso. Que a Força esteja com você.

Entrada

Cada caso de teste é composto por três linhas que indicam, respectivamente, as roupas sujas de Railson, do pai e da mãe dele, onde cada linha contém três números inteiros C, P e B, que representam a quantidade de roupas coloridas, pretas e brancas, separadas por um espaço em branco.

Saída

Para cada caso de teste, determine qual cor de peça tem maior quantidade, podendo ser “colorida”, “preta” ou “branca” são as possíveis mensagens exibidas, seguida de um espaço em branco e em qual nível a máquina vai funcionar, sendo 1, 2 ou 3 as possíveis saídas determinadas em crescente aos níveis da máquina.

Exemplos de Entrada

Entrada	Saída
12 2 3 6 5 2 5 3 1	colorida 2

Entrada	Saída
5 6 8 3 1 2 1 3 2	branca 1

Problema C. Livro de Caras

Nome do Programa: C.(c | pas | cpp | java | py)
Tempo: 1 segundo

A rede social Livro de Caras pretende influenciar pessoas nas próximas eleições dos países da América do Sul. Para isto, sua equipe de desenvolvimento pretende postar vídeos e espera que as pessoas curtam estes vídeos para que eles se propaguem na rede. Quando um usuário da Livro de Caras curte um vídeo, ele aparece para todos os amigos dele na rede social, aumentando assim o seu alcance. A equipe da Livro de Caras percebeu que o alcance do vídeo é maior quando as pessoas têm os mesmos interesses, pois elas "curtem" o vídeo e este aparece para os seus amigos. Isso quer dizer que, mesmo que uma rede contenha **João →Pedro →Maria →Daniela** e que João, Pedro e Daniela gostem de cachorros, se Maria não gosta de cachorros o alcance máximo (provável) de um vídeo sobre cachorros na rede, começando por João, será João, Pedro e Maria. Contudo, se João, Pedro, e Maria gostam de gatos, o alcance máximo de um vídeo sobre gatos começando em João será João, Pedro, Maria e Daniela.
Marco Zueiro, o dono da Livro de Caras, deseja saber quais as preferências para vídeos com o maior alcance provável de visualizações a partir de um determinado usuário. Para isto, ele contratou você.

Entrada

A entrada de dados é dividida em três partes. Na primeira são listados todos os usuários da rede, um por linha, com seus respectivos gostos pessoais. Esta primeira parte termina com **FIM**. A segunda parte é formada pela lista de amigos de cada usuário, um usuário e seus amigos por linha. Esta segunda parte também termina com **FIM**. A terceira e última parte contém o nome de um usuário por linha.

Saída

Para cada usuário informado na última parte da entrada, o programa deve imprimir o "gosto pessoal" do usuário para o qual um vídeo terá o maior número provável de visualizações como também este número esperado de visualizações.

Exemplos de Entrada

Entrada	Saída
Ayron Cavalo Paulinho Cavalo Vaca Ota Gato Cachorro Galinha Algoritmo Vaca Engenharia Diana Gato Cachorro Erick Redes Cavalo Ruan Algoritmo MPB Surf Joao Gato Galinha Cavalo FIM Ayron Paulinho Paulinho Joao Ota Diana Ota Joao Paulinho Diana Erick Ruan Diana Joao Paulinho Ruan Ota Erick Ruan Ota Joao Paulinho Ota Diana Ruan Erick Ota FIM Ayron Joao Paulinho Ota Diana Ruan Erick	Ayron Cavalo 5 Joao Cavalo 4 Paulinho Vaca 6 Ota Engenharia 6 Diana Cachorro 6 Ruan Surf 3 Erick Cavalo 3

Problema D. Jogo da Velha

Nome do Programa: D.(c | pas | cpp | java | py)
Tempo: 1 segundo

Implemente um programa capaz de dizer quem será o vencedor ou se haverá empate em uma partida do jogo da velha. O programa assume que os jogadores não cometem erros na estratégia de marcação.

Entrada

O programa recebe quem começou a partida e o estado do tabuleiro, uma partida por linha, na forma de uma cadeia de caracteres. A primeira posição indica quem começou a partida. Pode ser "X" ou "O". Em seguida um espaço em branco. Por fim, 9 posições com o estado do tabuleiro.
As posições podem conter um "X", "O" ou "-" (para indicar que ainda não foi marcada). Os três primeiros caracteres correspondem à primeira linha do tabuleiro, os três caracteres seguintes à linha do meio e os três últimos à última linha. A última linha da entrada de dados é uma linha em branco.

Saída

O programa deve imprimir "X GANHA" se X vai ganhar o jogo, "O GANHA" se O será vitorioso ou "EMPATE" se o jogo terminará empatado.

Exemplos de Entrada

Entrada	Saída
X --X------ O ----O---- X X--XXOOXO X X-X-XOOXO X X-XXOO-XO X XX-XO-OXO O OO-X----- X X---X--OO	EMPATE EMPATE O GANHA EMPATE X GANHA O GANHA O GANHA X GANHA

Problema E. Clock Tree Synthesis

Nome do Programa: E.(c | pas | cpp | java | py)
Tempo: 1 segundo

O processo de construção de um processador moderno envolve uma etapa chamada de clock tree synthesis. Nesta etapa, após os flip-flops serem posicionados, uma ligação entre os conectores do sinal de relógio deve ser feita, criando assim um caminho de material condutor. É desejável que a ligação conecte-os diretamente, mas nem sempre este é o caminho de menor "custo" e, nesses casos, a ligação pode passar antes por pinos de clock de outros flip-flops. Para cada ligação direta, existe um custo associado à sua construção, como também às ligações que já conectam outros flip-flops (devido a corrente suportada e resistência elétrica). Quando utilizando conexões dos outros flip-flops, isto pode distanciar o sinal de relógio do flip-flop de destino, por esse (e outros motivos) existe um limite máximo para o número dessas conexões que se pode utilizar.
Os flip-flops também foram ordenados segundo a preferência para conexão, baseada na largura das trilhas da árvore de clock. Assim, o sinal de clock do flip-flop de índice 1 é o que se prefere reutilizar em uma conexão (caso permitido), seguido pelo flip-flop 2 e assim por diante.

Você é o encarregado de escrever o algoritmo de construção da melhor árvore de clock. Para isso, são dados os flip-flops de origem, de destino e o número t de flip-flops cuja trilha de clock pode ser compartilhada. Seu programa deve encontrar o caminho de custo mínimo para a ligação dos pinos do sinal de clock dos flip-flops de origem e destino e que utilize as mesmas conexões de um máximo de t flip-flops. Por exemplo, se t = 3 o programa deve decidir se o menor custo é obtido na ligação direta, ou reutilizando o sinal de clock do flip-flop 1, ou 1 e 2 ou, no máximo, dos flip-flops 1, 2 e 3.

Entrada

A entrada é composta de diversos blocos da CPU. A primeira linha de cada bloco consiste em dois inteiro n, onde ($2 \leq n \leq 100$) e m, onde ($0 \leq m \leq 100000$), indicando o número de flip-flops e o número de conexões entre eles. Nas m linhas seguintes, temos três inteiros, u, v e w ($1 \leq u \leq n$, $1 \leq v \leq n$ e $0 \leq w \leq 100$), indicando que existe uma conexão entre u e v com custo w. Em seguida, um inteiro c, onde ($1 \leq c \leq 10000$), indicando o número de consultas, e nas c linhas seguintes, três inteiros o, d e t, onde ($1 \leq o \leq n$, $1 \leq d \leq n$ e $1 \leq t \leq n$), onde o é o flip-flop de origem, d é o de destino e t indica que os flip-flops 1, 2, . . . t podem ser usados na criação da árvore de *clock*.

A entrada termina com fim de arquivo.

Saída

Para cada bloco, o programa deve imprimir um identificador Bloco k, onde k é o número do bloco atualmente em análise. Para cada consulta, na ordem de entrada, o programa deve imprimir o custo mínimo ou −1, caso não exista um caminho entre os dois flip-flops.

Após cada bloco o programa deve imprimir uma linha em branco.

Exemplos de Entrada

Entrada	Saída
4 5 4 1 0 2 1 3 2 3 15 4 2 1 3 1 21 3 2 1 0 4 2 2 4 3 1 5 7 4 5 2 2 1 4 2 4 7 5 2 1 4 1 2 5 4 4 5 3 7 4 2 5 0 3 4 5 4 5 1 2 3 2	Bloco 1 3 Bloco 2 3 1 21 Bloco 3 1 9 2 -1

Problema F. Mochileiro

Nome do Programa: F.(c | pas | cpp | java | py)
Tempo: 1 segundo

Certo mochileiro pretende conhecer todo o país andando de carona. A estratégia utilizada pelo mochileiro é viajar pelas cidades pedindo carona até atingir a cota máxima de caronas antes que ele faça um descanso mais prolongado. Após descansar, o mochileiro define qual o novo número máximo de caronas que ele vai pedir até parar para um novo descanso. De posse de um mapa rodoviário, a cidade atual onde ele se encontra e a cota máxima de caronas entre cidades vizinhas, ajude o mochileiro com um programa que informa todas as cidades que podem ser alcançadas com essas restrições.

Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de teste contém quatro números inteiros C, E, L e P . Os valores C e E indicam respectivamente o número de cidades e estradas existentes. As cidades são identificadas por inteiros de 1 a C. Os valores L e P indicam, respectivamente, a cidade onde o mochileiro se encontra e o número máximo de caronas que ele planeja pegar. As E linhas seguintes contém, cada uma, a informação de uma estrada, representada por um par de números inteiros positivos X e Y , indicando que há uma estrada da cidade X para a cidade Y . O final da entrada é indicado por C = E = L = P = 0.

Saída

Para cada conjunto de teste de entrada, o programa deve imprimir três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato "Teste n", onde n é numerado a partir de 1. Na segunda linha devem aparecer os identificadores das cidades que podem ser alcançadas, em ordem crescente, separados por um espaço em branco. A terceira linha deve ser deixada em branco. A saída deve obedecer a formatação da seção 1.3.2.

Exemplos de Entrada

Entrada	Saída
5 4 2 1 1 2 2 3 3 4 4 5 9 12 1 2 2 1 1 5 2 1 3 2 9 3 3 4 4 8 4 7 7 6 5 6 4 5 3 7 0 0 0 0	Teste 1 1 3 Teste 2 2 3 4 5 6

Problema G. (2/3/4)-D Sqr/Rects/Cubes/Boxes?

Nome do Programa: G.(c | pas | cpp | java | py)
Tempo: 2 segundos

Veja a grade (4 × 4) abaixo. Você consegue dizer quantos quadrados e retângulos ela contém? Você pode assumir que quadrados não são retângulos. Talvez seja possível contar a mão, mas você conseguiria fazer isso para uma grade (100×100) ou para uma grade (10000×10000)? E para dimensões maiores? Isto é, você poderia contar quantos cubos ou caixas de tamanhos diferentes há em um cubo de dimensões (10 × 10 × 10) ou quantos hipercubos ou hipercaixas de tamanhos diferentes existem em um hipercubo de 4 dimensões de tamanho (5 × 5 × 5 × 5)? Lembre-se que o seu programa precisa ser muito eficiente (programas com respostas previamente armazenadas serão desconsiderados). Você pode assumir que quadrados não são retângulos, cubos não são caixas e hipercubos não são hipercaixas.

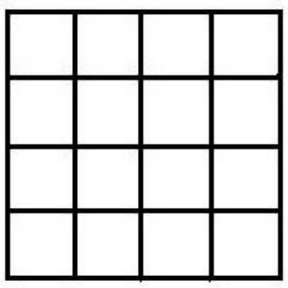

Fig: A 4x4 Grid

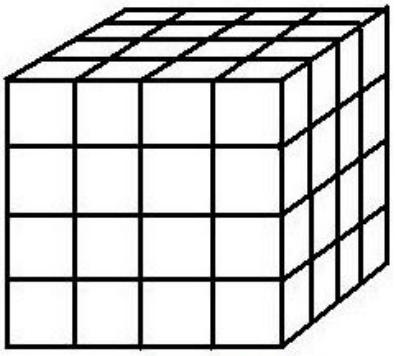

Fig: A 4x4x4 Cube

Entrada

A entrada contém um número inteiro N , onde (0 ≤ N ≤ 100) em cada linha, que é o comprimento de um lado da grade ou cubo ou hipercubo. Tal como para o exemplo acima, o valor de N é 4.
Pode haver mais de 100 linhas de entrada.

Saída

Para cada linha de entrada, deve ser gerada um saída com seis inteiros S2, R2, S3, R3, S4, R4 em uma única linha, onde S2 indica o número de quadrados contidos na grade bidimensional (N × N), R2 indica o número de retângulos contidos na grade bidimensional (N × N). S3, R3, S4, R4 tem o mesmo significado porém em dimensões superiores, como descrito antes.

Exemplos de Entrada

Entrada	Saída
1 2 3	1 0 1 0 1 0 5 4 9 18 17 64 14 22 36 180 98 1198