Seletiva Interna da UFMA 2010

Departamento de Informática – 15 de Julho de 2010

Problema A

Time de Basquete

Nome do arquivo fonte: **basquete.{c, cpp** ou **java}**

Zezinho foi convidado para treinar um time de basquete. Acontece que tudo que ele sabe sobre o jogo é que um time é formado por cinco titulares. Ele agora precisa criar algum critério para selecionar seus cinco jogadores. Como basquete é um jogo para pessoas altas, resolveu simplificar o processo de seleção: ele vai escolher os cinco jogadores mais altos para formar seu time, independente da habilidade deles. Você é o auxiliar de Zezinho. Sua tarefa é criar um programa de computador que lê como entrada a altura de N (4<N<201) jogadores e informa quais foram os cinco jogadores selecionados para o time. Quando houver mais de um jogador com a mesma altura, você deve selecionar para o time o primeiro que aparece na entrada (privilegie os que vieram para a peneira primeiro, eles devem ser mais interessados!).

Entrada

A entrada consiste de vários casos de teste. Cada caso de teste representa um conjunto de N jogadores (4<N<201) que vieram participar da peneira para entrar no time de basquete de Zezinho. Um caso de teste é composto por uma linha isolada com um inteiro N (o número de jogadores), seguido de N linhas, cada uma delas contendo a altura H em centímetros (150<H<230) do i-ésimo jogador que veio à peneira. O primeiro jogador que veio à peneira deve ser representado na saída pelo inteiro 1 e corresponde ao primeiro que aparece na entrada. Ele é seguido do jogador 2, que é o segundo a aparecer na entrada e assim por diante. Relembrando: se dois ou mais jogadores possuírem a mesma altura, o primeiro que aparece na entrada deve ser escolhido em detrimento daqueles que aparecem depois e possuírem a mesma altura dele. O final da entrada é indicado por um caso de testes em que o valor de N é igual a 0 (ZERO), o qual não deve ser processado e deve terminar o programa imediatamente. **Leia da entrada padrão.**

Saída

Para cada caso de testes na entrada, seu programa deve imprimir uma linha na saída contendo cinco inteiros S (1<=S<=N) separados por um único espaço em branco (não há um espaço em branco ao final da linha). O primeiro inteiro que aparecer na saída deve ser aquele que representa o jogador mais alto, seguido do segundo mais alto e assim por diante até o quinto inteiro. Note que se há dois ou mais jogadores com a mesma altura na peneira, aquele que veio primeiro na entrada deve ser considerado o mais alto. **Exiba na saída padrão, sem espaço em branco no final da linha.**

Exemplo de entrada	Respectiva saída
7 182 201 201 182 182 201 203 5 178 178 178 178 178 0	7 2 3 6 1 1 2 3 4 5

Autor: Prof. Carlos de Salles.

Seletiva Interna da UFMA 2010

Departamento de Informática – 15 de Julho de 2010

Problema B

Pilhas de Caixas

Nome do arquivo fonte: **pilhas.{c, cpp** ou **java}**

Num porto, várias caixas são empilhadas umas sobre as outras num ritmo frenético para embarcar nos navios. Acontece que algumas dessas pilhas ficam bem instáveis e eventualmente caem, causado prejuízo e perda de mercadorias. O chefe do porto precisa de um programa que indique para ele quando uma pilha de caixas é estável ou instável, uma vez que fazer isso manualmente seria proibitivamente cansativo.

Dada uma pilha de caixas informada como entrada, sua tarefa é indicar "sim" quando aquela pilha de caixas estiver construída de forma estável ou então "nao" (sem til), quando aquela pilha de caixas estiver prestes a cair. Uma pilha está estável se e somente se todas as caixas posicionadas acima de uma caixa qualquer da pilha forem MENORES OU IGUAIS àquela caixa. Note que uma pilha contendo uma única caixa é, por definição, sempre estável.

Entrada

A entrada é composta de várias pilhas de caixas. Uma pilha de caixas é representada na entrada por um inteiro N (1<=N<=10.000) em uma linha isolada, seguido de N linhas com um inteiro em cada, informando o tamanho de cada uma das N caixas. O tamanho de uma caixa é representado por um inteiro variando de 1 a 100. Esse inteiro é um código para a caixa de tal forma que, quanto maior é o código, maior é a caixa. Em outras palavras, a caixa de tamanho 1 é menor que todas as demais, enquanto a caixa de tamanho 100 é a maior possível. Por sua vez, obviamente, todas as caixas de mesmo código (ou tamanho) são exatamente iguais. Quando o valor de N é informado igual a 0 (ZERO), o programa deve terminar imediatamente sem processar essa entrada. **A entrada deve ser lida da entrada padrão.**

Saída

Para cada pilha de caixas lida na entrada, seu programa deve imprimir uma única linha na saída informando "sim" caso aquela seja uma pilha estável ou "nao", caso contrário. **A saída deve ser impressa na saída padrão.**

Exemplo de entrada	Respectiva saída
1 5 3 4 9 2 4 67 68 68 79 2 3 1 0	sim nao sim nao

Autor: Prof. Carlos de Salles.

Seletiva Interna da UFMA 2010

Departamento de Informática – 15 de Julho de 2010

Problema C

Anotações de Vinícius VI

Nome do arquivo fonte: **vinicius.{c, cpp** ou **java}**

Nós sabemos: todo ano tem essa dita cuja questão do Vinícius! E sempre há poucos heróis que a resolvem. Vinícius e seu padrinho estão querendo ficar mais populares. Por essa razão, eles bolaram algo bem interessante para esse ano.

Vinícius descobriu que todas as espécies do reino animal podem ser divididas em dois tipos, que ele chama de **fracos** e **fortes**. Qualquer animal **forte** é capaz de vencer qualquer animal **fraco** em combate voraz. Vinícius descobriu que os **fortes** NUNCA combatem entre si (malandros, esses fortes, heim?). Os **fortes** só combatem com **fracos.** E, pasme, os **fracos** desejariam não combater com ninguém (você acha que eles aceitam voluntariamente um combate com um forte?) porque eles acreditam que seriam vencidos por qualquer outro animal (é por isso que temos animais vegetarianos!). Bom … Vinícius alerta que se o conjunto de fracos não for unitário, estamos diante de um problema sério de lógica caso o pensamento dos fracos for verdadeiro: como pode qualquer fraco ser vencido por qualquer fraco?

Voltemos ao seu trabalho. Vinícius observou a natureza por um tempo razoável e anotou em seu inseparável caderno todos os combates que pôde presenciar. Se ele via um combate entre um gato e um rato, por exemplo, ia direto para o caderno. Ver um combate entre um gato e um rato dá a Vinícius duas informações relevantes e diretas: i) o gato e o rato não podem ser ambos **fortes**; e ii) o gato e o rato não podem ser ambos **fracos** (ora, um tem que ser forte e o outro fraco). O problema é que ele não anotou em seu caderno quem venceu cada combate, portanto não sabe dizer quem são os fortes e quem são os fracos. Para piorar, Vinícius se confundiu e fez algumas anotações erradas, escrevendo em seu caderno combates que nunca existiram.

Dadas as anotações de Vinícius, sua tarefa é informar "sim" caso as anotações de Vinícius fizerem sentido e for possível dividir os animais listados em dois conjuntos de tal forma que todos os combates sejam feitos entre membros de grupos diferentes. Caso haja erro (Vinícius fez uma anotação errada) e não houver nenhuma forma de dividir os animais listados em dois conjuntos de **fortes** e **fracos**, seu programa deve informar "nao".

Entrada

A entrada é formada por vários casos de testes, cada um deles listando vários combates entre pares de animais. A primeira linha de todo caso de testes inicia por um inteiro N (1<=N<=100). Seguem N linhas com o nome de dois animais em cada linha representando um combate entre aqueles dois animais (sim, é possível que Vinícius tenha anotado mais de uma vez o combate entre os mesmos dois animais). Vinícius nunca anotou um combate entre um animal e ele mesmo (dois animais da mesma espécie). O nome de um animal é uma cadeia de 1 a 15 letras a-z (minúsculas). **Leia da entrada padrão.**

Saída

Para cada caso de testes na entrada, imprima em uma linha isolada "sim", caso seja possível dividir em dois conjuntos de **fortes** e **fracos** todos os animais listados de tal forma que todos os combates daquele caso de testes se dê entre um animal de um conjunto com um animal do outro conjunto. Em caso contrário, caso não seja possível, imprima "nao" (sem til). **Imprima na saída padrão.**

Exemplo de entrada	Respectiva saída
2 gato rato leao boi 4 peixepalhaco sardinha tubarao peixepalhaco peixepalhaco tubarao tubarao sardinha 0	sim nao

Autor: Prof. Carlos de Salles, padrinho de Vinícius.

Seletiva Interna da UFMA 2010

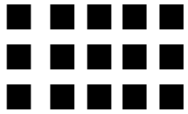
Departamento de Informática – 15 de Julho de 2010

Problema D

Comendo o Bolo

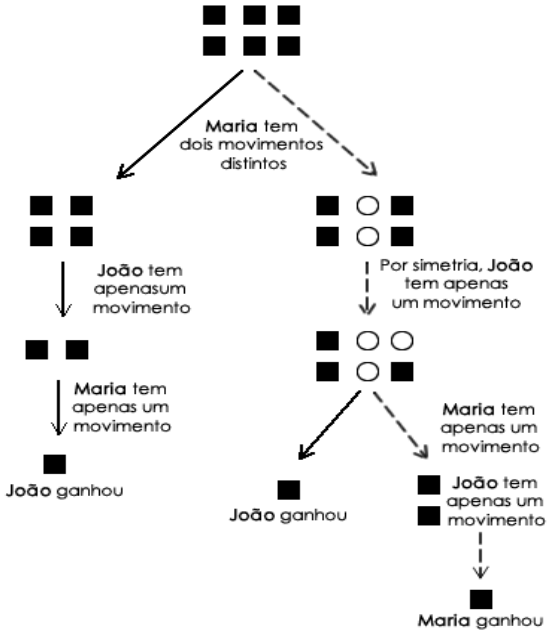
Nome do arquivo fonte: **bolo.{c, cpp** ou **java}**

Maria e João são dois garotos que gostam muito de brincar e adoram ir para festas de aniversário, entre outras coisas, porque gostam muito de bolo. No último aniversário que foram, a mãe de Maria apresentou uma nova brincadeira aos dois, chamada de **Comendo o Bolo**! Como já devem ter notado, essa brincadeira fez muito sucesso entre as crianças! A brincadeira consiste em dividir o bolo em L linhas e C colunas (formando L*C pedaços), conforme mostrado na figura ao lado (no caso, L=3 linhas e C=5 colunas):



O jogo é disputado por dois participantes em turnos. Maria joga primeiro, comendo uma **coluna** inteira de pedaços do bolo. Depois disso, o bolo pode ter se dividido em dois ou apenas diminuído de tamanho (quando a primeira ou última coluna for comida). Agora é a vez de João, que sempre deve comer uma **linha** inteira de pedaços do bolo. O jogo acaba quando é comido o último pedaço do bolo, quando é declarado o vencedor o participante que **COMEU** o último pedaço.

Embora os dois sejam excelentes comedores de bolo, eles não são tão bons com estratégias. João, em sendo sempre o segundo a jogar, quase sempre perdia o jogo. Ele ficava muito feliz em jogar (afinal estava comendo), mas não gostava muito de perder. De tanto perder, começou a desconfiar do jogo e resolveu pedir ajuda a você.



Sua tarefa é: dados o número de linhas e colunas no qual o bolo está dividido, informar qual dos dois jogadores é o vencedor do jogo, assumindo que **AMBOS** jogam da melhor forma possível para vencer. A Figura a seguir exemplifica configurações de jogadas possíveis para um bolo dividido em 2 linhas e 3 colunas. Nesse exemplo, Maria sempre vence o jogo escolhendo as jogadas certas (seguindo as setas em tracejado).

Entrada & Saída

A entrada é composta de vários casos de teste. Cada caso de testes aparece em uma linha isolada na entrada e é composto por dois inteiros L e C (1<=L<=6 e 1<=C<=6) separados por um único espaço em branco. Quando L e C forem ambos informados iguais a 0 (zero), o programa deve terminar imediatamente sem processar aquela entrada. A saída deve ser "joao" (sem til) caso João seja invariavelmente o vencedor daquele caso de testes ou "maria", caso Maria seja a vencedora.

Exemplo de entrada	Respectiva saída
1 2 2 3 0 0	joao maria

Autor: Roberto Gerson, membro da Equipe -, finalista da Maratona de Programação SBC 2006.

Fonte do jogo: http://www.madras.fife.sch.uk/maths/games/eatcake.html

Seletiva Interna da UFMA 2010

Departamento de Informática – 15 de Julho de 2010

Problema E

Saltando o muro

Nome do arquivo fonte: **muro.{c, cpp** ou **java}**

Em uma gincana existe uma prova denominada *Saltar o Muro,* que envolve habilidades atléticas e matemáticas. Como o próprio nome já diz, a tarefa é simples: um membro selecionado pela equipe deve saltar o muro. Entretanto, antes dele saltar, a equipe pode escolher um número fixo de blocos e empilhar TODOS os blocos próximo ao muro, para ajudá-lo. Escolher uma quantidade incorreta de blocos pode resultar que a equipe não consiga realizar a tarefa por ter poucos blocos de apoio – deixando o muro ainda muito alto – ou ainda a equipe perca a prova por ter blocos demais – o que acaba tomando mais tempo para empilhá-los do que o necessário.

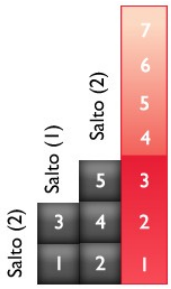


Figura 1: Péssima escolha de quantidade de blocos

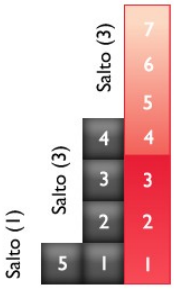


Figura 2: Boa escolha de quantidade de blocos

Na disputa de *Saltar o Muro* do ano passado, por exemplo, o muro era de 7 **ua** (unidades de altura). O atleta da equipe NAOPULA conseguia saltar 2 **ua** e o especialista em cálculo da equipe solicitou 5 blocos, formando então um cenário impossível para a realização da tarefa (**Figura 1**). Nesse mesmo ano, a equipe vencedora, a PULAPULA, tinha um atleta que conseguia saltar 3 **ua** e o seu especialista em cálculo também escolheu 5 blocos, conseguindo organizá-los de uma forma eficiente. (**Figura 2**).

A sua equipe vem se preparando já há alguns meses para a gincana deste ano, e está muito entusiasmada. Na última hora, entretanto, o atleta de sua equipe se machucou. Sem tempo para treinos, você deve confiar na capacidade do atleta substituto e sugerir o número certo de blocos para qualquer que seja a altura em **ua** que ele afirma ser capaz de saltar. Para agilizar a escolha dos blocos para a sua equipe, você resolveu utilizar seus conhecimentos em programação para desenvolver um programa que resolva isso para você. Seu programa deve ser capaz de calcular o número mínimo de blocos auxiliares que o atleta, com capacidade de salto igual a **S**, necessita para *Saltar o Muro* de altura **H**.

Entrada

A entrada é composta por vários casos de teste. Cada caso de teste é formado pelo par de inteiros H (1<=H<=1000) e S (1<=S<=1000), representando respectivamente a altura do muro e a capacidade de salto do atleta. Quando o valor da altura (H) e da capacidade do salto (S) for informados iguais a 0 (ZERO), o programa deve terminar imediatamente sem processar essa entrada. **A entrada deve ser lida da entrada padrão.**

Saída

Para cada altura e capacidade de salto informada na entrada, seu programa deve imprimir na saída o valor mínimo de blocos necessário para que o atleta consiga saltar o muro.

Exemplo de entrada	Respectiva saída
7 3 3 1 10 4 0 0	5 3 8

Autor: Eduardo Cruz Araújo, membro da Equipe Diott, finalista da Maratona de Programação da SBC em 2007.

Seletiva Interna da UFMA 2010

Departamento de Informática – 15 de Julho de 2010

Problema F

Os IGNORANTIUNs

Nome do arquivo fonte: **ignora.{c, cpp** ou **java}**

Os SABIDEINFs são um povo bem pacato e tido como os mais sábios do universo. Mesmo a contragosto, entretanto, eles tiverem que entrar em uma batalha de vida ou morte com o povo dos IGNORANTIUNs, já que com esses últimos não tem conversa e eles querem invadir o planeta pacato e em progresso dos SABIDEINFs. Os SABIDEINFs, fazendo jus ao fato de serem pacatos e muito sábios, inventaram uma nova arma que permite ganhar a guerra sem derramar sangue de nenhum dos IGNORANTIUNs! Tudo que essa arma não-mortal mas extremamente poderosa faz é: ao acertar uma nave, ela a desintegra e teletransporta todos os seus integrantes de volta ao seu planeta de origem.

Todas as naves dos IGNORANTIUNs são circulares e o seu centro está em uma posicao (Xn, Yn) do universo. Sim, o universo aqui é representado por um plano bidimensional. A nave possui um raio de Rn unidades. Os tiros dessa super-arma sempre saem do ponto de origem (0,0) em linha reta, em direção a um ponto (Xm, Ym) - para o qual a arma está mirando - e possuem um alcance máximo de D unidades.

Dados a posição de uma das naves e um tiro executado pela arma dos SABIDEINFs, será que você consegue fazer um programa que informa se essa nave foi atingida por esse tiro, e em caso afirmativo, informe em qual ponto ela foi atingida?

Entrada

A entrada é formada por vários casos de testes. Cada caso de testes é composto por duas linhas. A primeira linha contém três inteiros -1000 <= Xn, Yn <= 1000 e 1 <= Rn <=1000, informando, respectivamente, o ponto (Xn, Yn) onde o centro da nave dos IGNORANTIUNs está localizado e o raio (Rn) dessa nave. A segunda linha também contém três inteiros -1000 <= Xm, Ym <= 1000 e 1<=D<=1000, informando o ponto (Xm, Ym) para o qual a arma está apontando e o alcance máximo (D) que ela pode atingir. O ponto (Xm, Ym) nunca será informado igual a (0, 0). Adicionalmente, nunca a nave conterá o ponto de disparo (0,0) em seu interior. Quando Xn, Yn e Rn forem informados iguais a 0 (zero), seu programa deve ser interrompido e ignorar essa entrada. **Leia da entrada padrão.**

Saída

Para cada caso da entrada, seu programa deve produzir uma linha contendo "nao" (sem til) caso o tiro não acerte a nave ou "sim", seguidos por dois números reais Xi e Yi informando o ponto de impacto do tiro com a nave. Os números Xi e Yi devem ser impressos com duas casas decimais. **Imprima na saída padrão.**

Exemplo de entrada	Respectiva saída
3 0 1 5 0 7 3 0 1 0 5 10 2 2 1 4 2 50 2 2 1 4 2 1 0 0 0	sim 2.00 0.00 nao sim 2.00 1.00 nao

Autor: Roberto Gerson.