Universidade Federal do Maranhão
Departamento de Informática

Maratona de Programação
Inicinate 2009.1

27 de Março de 2009

(Este caderno contém 7 problemas; as páginas estão numeradas de 1 a 8 não contando esta página de rosto)

Realização:

Apoio:

Patrocínio:

Maratona de Programação Iniciante 2009.1

Departamento de Informática
27 de Março de 2009

Problema A

Problema dos Reis

Arquivo fonte: reis.c, reis.cpp, reis.java ou reis.pas

Um problema bem conhecido entre os estudantes e professores da área de computação é o problema das 8 rainhas. Os primeiros relatos deste problema datam do início do século, estudado por Gauss e bem conhecido dos livros que tratam de combinatória ou matemática recreativa.

O problema consiste em dispor oito 8 rainhas em um tabuleiro 8x8 de tal modo que elas não se ataquem.

Baseado neste problema, Joãozinho resolveu inventar um novo problema: O Problema dos Reis. Neste novo problema, Joãozinho está interessado em saber quantos reis, no máximo, é possível colocar em um tabuleiro de forma que nenhum rei ataque o outro.

Obs: Um rei ataca somente as casas vizinhas à qual ele se encontra.

Entrada

A entrada é composta por vários casos de teste. Cada caso de teste inicia-se com uma linha contendo dois inteiros **L** e **C** que representam as dimensões (largura e comprimento) do tabuleiro. A entrada termina quando **L = 0** e **C = 0,** a qual não deve ter uma saída correspondente. **Leia da entrada padrão.**

Limites: 0 <= **L** <= 10000, 0 <= **C** <= 10000.

Saída

Para cada caso de teste da entrada imprima uma linha na saída que contenha o número máximo de reis que podem ser colocados neste tabuleiro de forma que nenhum ataque um outro. **Use a saída padrão.**

Exemplo de Entrada	**Respectiva Saída**
3 3 10 10 0 0	4 25

Autor do Problema: Roberto Azevedo, finalista da Maratona de Programação da SBC 2006.

Problema B
Procurando a saída

Arquivo fonte: saida.c, saida.cpp, saida.java ou saida.pas

Chiquinho encontra-se na frente de um muro infinito para ambos os lados e deseja encontrar uma porta neste muro. Chiquinho sabe que existe uma porta e que esta está a **X** passos de distância da sua posição original. Só que tem um problema, além dele não saber o valor de **X**, também não sabe se a porta está à sua direita ou à sua esquerda. Sendo assim, ele decide adotar uma estratégia para encontrar a porta.

Sua estratégia é andar um passo para a direita, depois dois para a esquerda, depois três a direita, e assim sucessivamente, até encontrar a porta. Ele não sabe exatamente se esta é uma boa estratégia, então decide chamar você para ajudá-lo.

Faça um programa que dado a distância **X** que a porta está da posição inicial de Chiquinho, informe quantos passos serão necessários para encontrá-la, caso ela esteja à direita e quantos passos serão necessários caso ela esteja à esquerda.

Entrada
A entrada é composta por vários casos de teste. Cada caso de teste consiste de uma linha contendo um inteiro **X** que representa a quantos passos da posição inicial de Chiquinho a porta está. A entrada termina quando **X = 0**. **Leia da entrada padrão.**

Limites: 0 <= **X** <= 2000.

Saída
Para cada caso de teste da entrada, imprima uma linha na saída informando, respectivamente, quantos passos Chiquinho deverá dar (seguindo a sua estratégia) se a porta estiver à direita e quantos passos serão necessários caso à porta esteja à esquerda. **Use a saída padrão.**

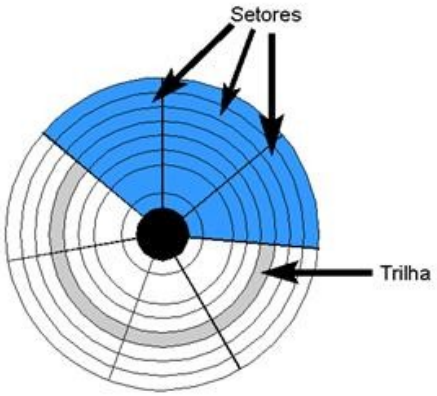
Exemplo de Entrada	**Respectiva Saída**
2 5 6 0	6 10 45 55 66 78

Autor do Problema: Roberto Azevedo, finalista da Maratona de Programação da SBC 2006.

Problema C
Setores no Disco

Arquivo fonte: disco.c, disco.cpp, disco.java ou disco.pas

Vocês se lembram como aquele velho LP que vocês tinham em casa funcinava? Ele usava um cabeçote posicionado em um certo lugar no disco para ler uma informação. É incrível como a maioria de dispositivos de armazenamento(Discos Rígidos, CDs, DVDs) ainda usa a mesma idéia.



Entretanto, há diferenças na forma de armazenamento da informação. No seu HD por exemplo, cada disco é dividido em trilhas, como se fosse cada circunferência concêntrica ao disco. E cada trilha é dividida em setores, que são uma fração dessa trilha. O engraçado é que os setores de uma trilha não são numerados na ordem em que eles estão dispostos. O motivo disso é porque em cada setor é lido de uma só vez, mas entre acabar uma leitura e começar outra existe um intervalo. Enquanto esse tempo passa, o disco continua girando e passa por alguns setores até voltar a ler de novo. Por exemplo, um disco tem um atraso de um setor pra voltar a ler a trilha e esta tem 7 setores. Se os setores estivessem organizados em ordem, e queremos ler o setor 1 e 2, leríamos o 1, o disco não leria o 2 porque está no intervalo e só voltaria a ler quando estivesse no setor 3. Nesse exemplo, ele teria que esperar uma volta no disco, mas se o disco fosse organizado da forma 1, 5, 2, 6, 3, 7, 4, quando o disco voltasse a ler, ele estaria exatamente no setor 2.

Você foi contratado por uma empresa para fazer uma rotina no seu Sistema Operacional que numera os setores automaticamente.

Entrada
No começo do programa será informado um inteiro que é a quantidade casos de teste. Para cada caso serão informados 2 números inteiros representando respectivamente o tamanho da trilha e o atraso(em número de setores) de voltar a ler, em que ambos são estritamente positivos menores que 50. **Leia da entrada padrão.**

Saída
A saída deve ser a melhor forma de se organizar o disco começando do setor 1. Veja o exemplo de saída. **Use a saída padrão.**

Exemplo de Entrada	**Respectiva Saída**
5 7 1 8 1 8 2 7 8 10 10	 1 5 2 6 3 7 4 1 5 2 6 3 7 4 8 1 4 7 2 5 8 3 6 1 5 2 6 3 7 4 1 2 3 4 5 6 7 8 9 10

Autor do Problema: Pablo Durans, finalista da Maratona de Programação da SBC 2007.

Problema D
Sitio do vovô

Arquivo fonte: sitio.c, sitio.cpp, sitio.java ou sitio.pas

O vovô Zé adorava seu neto. Sempre quando estava chegando o final do mês ele se preparava para receber seu netinho. Ele comprava um presente e pesquisava uma nova brincadeira. Na verdade, a brincadeira era sempre a mesma, ele fazia um desafio e se antes de ir embora seu neto solucionasse o problema, o garoto poderia ir para casa com um grande presente.

Depois de acomodar seu querido neto no sítio, vovô Zé sentou-se na varanda com o garoto e começou a brincadeira dizendo: “Sei que amanhã é seu aniversário, então tenho uma nova proposta para você. Dessa vez vou lhe fazer várias perguntas e para cada resposta correta vai levar um presente diferente. Todas as perguntas vão estar relacionadas a um mesmo problema, por isso fique muito atento desta vez. Como você sabe, aqui no sítio nós só criamos galinhas e coelhos. Então ai vai a pergunta: Suponha que temos X cabeças de animais e Y números de pés dos mesmos, quantas galinha e coelhos temos no sítio? Bom, pense com cuidado e resolva o problema que amanhã será um dia de muitos presentes!”.

Joãozinho ficou muito feliz, um presente já era ótimo, agora ganhar o tanto que pudesse seria o melhor aniversário da sua vida. Mas depois de um tempo a ficha caiu. O problema era tão intrigante quanto o tamanho da sua felicidade. Ele então lembrou que certa vez um amigo lhe falou sobre um grupo de maratona de programação e como ele poderia ser útil na resolução de problemas que envolvam lógica. Assim, entrou na internet e relatou o seu problema ao grupo.

Amanhã é o aniversário de Joãozinho. Ajude-o a ter o melhor aniversário da sua vida e desenvolva um programa para responder as perguntas do avô.

Entrada
A entrada começa com um número 1 <= **N** <= 1000 , que representa o número de casos de teste. Então seguem **N** linhas, cada uma contendo a quantidade **X** de cabeças e a quantidade **Y** de pés de animais, em que 1 <= **X** <= 500, 2***X** <= **Y** <= 4***X** e **Y** mod 2 = 0. **Leia da entrada padrão.**

Saída
Para cada caso de teste da entrada, imprima uma linha na saída contendo, respectivamente, a quantidade de galinha e coelhos que existe no sítio. **Use a saída padrão.**

Exemplo de Entrada	**Respectiva Saída**
2 2 4 5 18	2 0 1 4

Autor do Problema: Eduardo Araújo, finalista da Maratona de Programação da SBC 2007.

Problema E
Contando as letras

Arquivo fonte: letras.c, letras.cpp, letras.java ou letras.pas

Um pesquisador linguístico está escrevendo um artigo sobre como as diferentes línguas baseadas no alfabeto romano tem sido utilizada desde sua origem. Em uma das seções do artigo o pesquisador queria estabelecer as diferença entre essas línguas. O primeiro índice a ser avaliado era a diferença da intensidade da utilização de vogais e consoantes entre elas.

O pesquisador então coletou vários textos escritos em diferentes línguas baseadas nesse alfabeto. Selecionou aleatoriamente vários trechos desses textos e para fazer a contagem das vogais e consoantes contactou o departamento de informática em busca de algum aluno que pudesse desenvolver um programa capaz de realizar essa contagem. Rapidamente nosso departamento respondeu e indicou você para ajuda o pesquisador.

Contribua com o trabalho de pesquisa e orgulhe o nosso departamento.

Entrada
A entrada começa com um número 1 <= **N** <= 1000 , que representa o número de casos de teste. Então seguem **N** linhas, cada uma com um trecho de máximo 80 caracteres. Cada trecho só pode ser formado por caracteres alfanuméricos, caracteres de pontuação ('?', '!', '.' e ',' apenas) e espaços em branco. As letras são sempre minúsculas e não há letras acentuadas. **Leia da entrada padrão.**

Saída
Para cada caso de teste da entrada, imprima uma linha na saída contendo, respectivamente, a quantidade de vogais e consoantes. **Use a saída padrão.**

Exemplo de Entrada	**Respectiva Saída**
2 ola mundo! hello world!	4 4 3 7

Autor do Problema: Eduardo Araújo, finalista da Maratona de Programação da SBC 2007.

Problema F
Lista de exercício

Arquivo fonte: lista.c, lista.cpp, lista.java ou lista.pas

As listas de exercício do professor Haroldo sempre escondem um detalhe especial. As questões são sempre associadas a sua numeração seguindo uma determinada lógica. Assim, não é por acaso que determinados problemas eram mais complexos que outros. Como sempre, ele pede que cada aluno escolha no mínimo 5 questões e avisa que quanto maior a quantidade de exercícios respondidos maior a quantidade de pontos extras na prova.

Dessa vez o professor Haroldo aprontou mais do que de costume. Sua intenção era que mesmo seus alunos escolhendo apenas 5 questões, eles acabem tendo que resolver muito mais que isso. Então, ele ligou os enunciados dos problemas de tal maneria que para resolver um problema **X** fosse necessário as respostas de até **Y** problemas.

Para dar chances aos seus alunos de descobrirem a melhor forma de escolher as questões, um dia antes de entregar a lista ele dá uma introdução básica da lógica que determina a dependência entre as questões. Então, através de exemplos o professor tenta passar o recado a eles. Ele começa dizendo que todas as questões dependem da 1ª. A partir daí, ele começa a dar uma bateria de exemplos:

- a 2ª depende da 1ª;
- a 4ª depende da 2ª e da 1ª;
- a 6ª depende da 3ª , 2ª e da 1ª;
- a 12ª depende da 6ª, 4ª, 3ª, 2ª e da 1ª;
- a 11ª depende apenas da 1ª;
- a 9ª depende da 3ª e da 1ª;
- a 15ª depende da 5ª, 3ª, e da 1ª.

Todos logo entendem que escolher as questões com a menor quantidade de dependências era a maneira mais fácil de ter todos os pontos extras na prova. Joãozinho estava muito atrasado em outras disciplinas e não podia perder muito tempo para garantir todos seus pontos extras. Ele precisa de um programa que dado uma lista de questões que ainda não foram escolhidas por seus amigos ele soubesse qual a ordem de escolha que lhe desse menor dor de cabeça, ou seja, a ordem que exigisse a resolução do menor número de problemas.

Para facilitar, não leve em consideração que determinas questões possam ter as mesmas dependências, ou seja, suponha que você realmente ira responder uma questão mais de uma vez. Em caso de empate, organize as questões na ordem crescente normal.

Entrada

A entrada começa com um número 1 <= **N** <= 1000 , que representa o quantidade de casos de teste. Então seguem **N** casos, cada um começando com uma linha contendo um número 5 <= **Q** <= 100 de questões que ainda sobraram para escolha. Na linha seguinte é listado então **Q** números que variam de **1** à **100,** que representam a numeração das questões. **Leia da entrada padrão.**

Saída

Para cada caso de teste, imprima a lista de questão na ordem mais fácil para que Joãozinho possa escolher suas questões. **Use a saída padrão.**

Exemplo de Entrada	**Respectiva Saída**
2 10 1 2 3 4 5 6 7 8 9 10 5 13 11 3 19 4	1 2 3 5 7 4 9 6 8 10 3 11 13 19 4

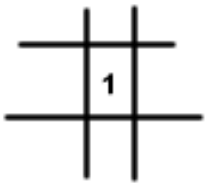
Autor do Problema: Eduardo Araújo, finalista da Maratona de Programação da SBC 2007.

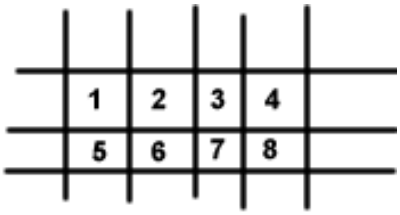
Problema G
Retângulos

Arquivo fonte: retangulos.c, retangulos.cpp, retangulos.java ou retangulos.pas

Como trabalho final da disciplina de Introdução à Informática, o professor resolveu passar um trabalho que envolvesse toda a turma. Ele começou a escrever várias perguntas estranhas no quadro e assim que terminou disse que o trabalho final era fazer uma biblioteca de algoritmos que respondessem todas aquelas peguntas. A verdadeira lição não era as questões em si, mas o trabalho em equipe e como a expressão "dividir para conquistar" deve fazer parte da vida de um cientista da computação. Apesar de claramente exigir um pouco de raciocínio, os alunos logo começaram a discutir sobre a utilidade de uma biblioteca para aquelas perguntas estranhas. Ignorando os comentários, o professor divide as questões entre os alunos e avisa que a biblioteca tem que estar completa para que todos possam ser aprovados.

José acabou pegando uma bem fácil: "Se traçarmos **X** linhas horizontais e **Y** linhas verticais, quantos retângulos formaremos?". Antes mesmo de ir para casa ele pegou uma folha de papel e começou a fazer alguns desenhos. Através deles, visualizar a quantidade de retângulo ficou muito fácil e logo conseguiu generalizar o caso para que pudesse desenvolver o algoritmo. Como não pudia falhar com seus amigos, ele tinha que ter certeza que seu algoritmos está funcionando. Então, preparou uma bateria de teste e pediu que você resolvesse o mesmo problema para que pudesse comparar seus resultados.





Ajude José e consequentemente sua turma a passar na disciplina.

Considere que as linhas nunca são coincidentes e que retângulos formados pela união de 2 ou mais retângulos não devem ser contados.

Entrada
A entrada começa com um número 1 <= **N** <= 100 , que representa a quantidade de casos de teste. Então seguem **N** linhas, cada uma contendo o número 0 <= **H** <= 100 de linha horizontais e um número 0 <= **V** <= 100 de linhas verticais. **Leia da entrada padrão.**

Saída
Para cada caso de teste da entrada, imprima uma linha na saída contendo a quantidade retângulos que foram formados pelas linhas. **Use a saída padrão.**

Exemplo de Entrada	**Respectiva Saída**
2 2 2 5 3	1 8

Autor do Problema: Eduardo Araújo, finalista da Maratona de Programação da SBC 2007.