OBI2013

Caderno de Tarefas

Modalidade **Programação** • Nível **1**, Fase **1**

18 de maio de 2013

A PROVA TEM DURAÇÃO DE **4** HORAS

Promoção:

Sociedade Brasileira de Computação

Patrocínio:

Fundação Carlos Chagas

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 6 páginas (não contando a folha de rosto), numeradas de 1 a 6. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta ao *help* do ambiente de programação se este estiver disponível.
- As tarefas têm o mesmo valor na correção.
- 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 primeiro as questões mais fáceis.
- 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 *.cc* ou *.cpp*; soluções na linguagem Pascal devem ser arquivos com sufixo *.pas*; 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 devem ser arquivos com sufixo *.py*. Para problemas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada problema.
- Ao final da prova, para cada solução que você queira submeter para correção, copie o arquivo fonte para o seu diretório de trabalho ou disquete, conforme especificado pelo seu professor.
- Não utilize arquivos para entrada ou saída. 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 entrada e saída de dados:
  - em Pascal: *readln, read, writeln, write*;
  - em C: *scanf, getchar, printf, putchar*;
  - em C++: as mesmas de 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: *read,readline,readlines,print,write*
- Procure resolver o problema de maneira eficiente. Na correção, eficiência também será levada em conta. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.

Corrida

Nome do arquivo fonte: `corrida.c`, `corrida.cpp`, `corrida.pas`, `corrida.java`, *ou* `corrida.py`

A Federação de Corridas de Charrete (FCC) organiza todo ano a Subida Brigite Cardoso (SBC), disputada nas ladeiras de paralelepípedo de Ouro Preto. A corrida é uma das mais tradicionais do esporte, completando 100 anos em 2013. Para comemorar o centenário, a FCC pretende integrar dispostivos GPS às charretes, permitindo aos espectadores desfrutarem de dados de telemetria em tempo real.

No mesmo viés de inovação tecnológica, a FCC transmitirá a SBC via satélite para todo o planeta, e quer integrar a telemetria na transmissão, indicando qual seria o vencedor da corrida se as charretes mantivessem suas velocidades até o final da corrida; ela pediu que você escrevesse um programa que, dados as distâncias até a linha de chegada, as velocidades e os números das duas charretes que lideram a corrida, determina quem seria o vencedor da corrida (você pode supor que as charretes não cruzam a linha de chegada simultaneamente).

Entrada

A entrada consiste de duas linhas; cada linha descreve uma das charretes que lidera a corrida. A descrição de uma charrete consiste de três inteiros N, D e V indicando, respectivamente, o número da charrete, a sua distância à linha de chegada em metros, e a sua velocidade, em quilômetros por hora. Os números das duas charretes são distintos.

Saída

Imprima uma única linha, contendo um único número inteiro, indicando o número da charrete que seria vencedora, conforme descrito acima.

Restrições

- $1 \leq N \leq 99$
- $0 < D \leq 1000$
- $0 < V \leq 50$

Exemplos

Entrada	**Saída**
`45 900 40` `17 300 20`	`17`

Entrada	**Saída**
`1 1000 100` `2 1000 99`	`1`

Capital

Nome do arquivo fonte: `capital.c`, `capital.cpp`, `capital.pas`, `capital.java`, *ou* `capital.py`

O governo do estado de Queensland está com problemas sérios de trânsito na capital Brisbane, onde estão os prédios administrativos. Para desafogar o trânsito, o prefeito de Brisbane e o governador de Queensland decidiram que uma nova capital administrativa deve ser construída em uma área fora de Brisbane. Para projetar a nova capital, o renomado arquiteto minimalista Joe Bloggs foi contratado.

Bloggs foi informado de que o terreno destinado à nova capital ainda não foi demarcado, mas será retangular. Além disso, a cidade deverá ser dividida em quatro zonas, uma delas destinada a uma reserva ambiental e cada uma das outras três receberá os novos prédios de cada um dos três poderes (Executivo, Legislativo e Judiciário). Em um arroubo de criatividade, Bloggs decidiu que duas avenidas, perpendiculares entre si, cada uma paralela a dois dos lados do terreno retangular, dividirão a capital nas quatro zonas.

Bloggs recebeu do governo as áreas de cada uma das zonas e, após muito esforço, encontrou um retângulo que pode ser dividido conforme seus planos e de forma a respeitar as áreas delimitadas. No entanto, a Fundação de Conservação dos Cangurus determinou que a área destinada à reserva ambiental era muito pequena, o que obrigou o governo a alterar as áreas das quatro zonas. Após receber as novas medidas, Bloggs tentou encontrar um novo retângulo que viabilizasse seu projeto, porém sem sucesso. Cansado de fazer testes, ele pensou que talvez tenha que abandonar sua brilhante ideia. Por isso, ele pediu para você escrever um programa que, dadas as áreas das quatro zonas, determine se ele poderá ou não manter seu projeto (ou seja, se existe um retângulo que possa ser dividido por duas retas perpendiculares, cada uma paralela a dois dos lados do retângulo, tal que as quatro áreas formadas obedeçam às exigências do governo).

Entrada

A entrada consiste de uma única linha contendo quatro inteiros A_1, A_2, A_3, A_4, indicando a área de casa uma das zonas.

Saída

Imprima uma única linha contendo um único caractere: '`S`' se Bloggs pode preservar seu projeto e '`N`' caso contrário.

Restrições

- $1 \leq A_i \leq 10^4$

Exemplos

Entrada	**Saída**
`1 2 4 8`	`S`

Entrada	**Saída**
`1 2 3 4`	`N`

Entrada	**Saída**
`15 14 6 35`	`S`

Rodovia

Nome do arquivo fonte: `rodovia.c`, `rodovia.cpp`, `rodovia.pas`, `rodovia.java`, *ou* `rodovia.py`

As estradas da Nlogônia estão severamente danificadas, devido ao intenso fluxo de veículos pesados criado pelo desenvolvimento econômico do reino. Para resolver o problema, o rei da Nlogônia decretou que seriam construídas novas rodovias. O decreto determinou que:

- todas as rodovias construídas terão mão única, e ligarão exatamente duas cidades;
- nenhum par de rodovias se intersectará — serão construídos viadutos, túneis e pontes conforme necessário;
- por razões orçamentárias, o número de rodovias a construir será igual ao número de cidades que existem na Nlogônia;
- deve ser possível, partindo de qualquer cidade, chegar a qualquer outra cidade usando só as novas rodovias, sempre respeitando a mão das rodovias.

O engenheiro-chefe do reino desenhou uma proposta de mapa viário; o rei verificou que o plano satisfaz as três primeiras restrições, mas não conseguiu verificar a última. Por isso, ele pediu que você escrevesse um programa que determina se o plano de rodovias permite viajar de qualquer cidade até qualquer outra cidade da Nlogônia.

Entrada

A primeira linha de cada caso de teste contém um inteiro N, indicando o número de cidades. Cada uma das N linhas seguintes descrevem uma estrada: a linha contém dois inteiros A e B que indicam que existe uma estrada de mão única ligando a cidade A a outra cidade, B (as cidades são numeradas de 1 a N).

Saída

Imprima uma única linha contendo um único caractere: '`S`' se for possível ir de qualquer cidade a qualquer outra cidade por rodovias e '`N`' caso contrário.

Restrições

- $2 \leq N \leq 10^4$
- $A \neq B$;

Informações sobre a pontuação

- em um conjunto de casos de teste totalizando 20 pontos, $N \leq 3$;
- em um conjunto de casos de teste totalizando 40 pontos, $N \leq 8$;

Exemplos

Entrada	**Saída**
3 1 2 2 3 3 1	S

Entrada	**Saída**
3 1 2 2 3 1 3	N

Entrada	**Saída**
6 1 2 2 3 4 1 5 6 3 5 6 4	S

Entrada	**Saída**
6 1 2 2 3 3 1 4 5 5 6 6 4	N

Robô

Nome do arquivo fonte: `robo.c`, `robo.cpp`, `robo.pas`, `robo.java`, *ou* `robo.py`

Um novo robô de limpeza para um grande salão retangular está sendo desenvolvido. O robô vai percorrer o caminho definido por uma linha marcada no chão, que é coberto com ladrilhos quadrados, brancos e pretos: ladrilhos pretos indicam o caminho que o robô deve percorrer. Ao movimentar-se, o robô pode andar apenas em linha reta, para a frente. Parado, o robô pode girar para as quatro direções (Norte, Sul, Leste e Oeste).

Dados um mapa indicando a cor de cada ladrilho no chão e a posição inicial do robô, você deve escrever um programa que determine a posição final do robô.

Entrada

A primeira linha contém dois inteiros L e C indicando as dimensões do salão (número de linhas e número de colunas), medidas em ladrilhos. A segunda linha contém dois inteiros A e B indicando respectivamente a linha e a coluna da posição inicial do robô (as linhas são numeradas de 1 a L, de cima para baixo; as colunas são numeradas de 1 a C, da esquerda para a direita). Cada uma das L linhas seguintes contém C inteiros, zeros ou uns. Nessa representação, o valor '`1`' indica que o ladrilho corresponte é preto. O ladrilho da linha A e coluna B sempre é preto. O caminho do robô é definido unicamente: em nenhum momento o robô necessita fazer uma escolha sobre em qual direção ir (em outras palavras, todo ladrilho preto tem no máximo dois vizinhos pretos e o ladrilho inicial tem um vizinho preto).

Saída

Seu programa deve imprimir apenas uma linha, contendo dois números inteiros, respectivamente a linha e a coluna da posição final do robô.

Restrições

- $1 \leq L, C \leq 1000$
- $1 \leq A \leq L$, $1 \leq B \leq C$
- A posição final é diferente da posição inicial.

Exemplos

Entrada	**Saída**

`3 5`
`1 1`
`1 0 0 0 1`
`1 0 0 1 1`
`1 1 1 1 0`
	`1 5`

Entrada	**Saída**

`4 7`
`3 4`
`0 0 1 1 1 1 1`
`1 1 1 0 0 0 1`
`1 0 0 1 0 1 1`
`1 1 0 1 1 1 0`
	`4 2`