OBI2012

Caderno de Tarefas

Modalidade **Programação** • Nível **2**, Fase **1**

31 de março de 2012

A PROVA TEM DURAÇÃO DE **5** HORAS

Promoção:

Sociedade Brasileira de Computação

Patrocínio:

Fundação Carlos Chagas

v1.0

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 9 páginas (não contando a folha de rosto), numeradas de 1 a 9. 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.

Colchão

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

João está comprando móveis novos para sua casa. Agora é a vez de comprar um colchão novo, de molas, para substituir o colchão velho. As portas de sua casa têm altura H e largura L e existe um colchão que está em promoção com dimensões $A \times B \times C$.

O colchão tem a forma de um paralelepípedo reto retângulo e João só consegue arrastá-lo através de uma porta com uma de suas faces paralelas ao chão, mas consegue virar e rotacionar o colchão antes de passar pela porta.

Entretanto, de nada adianta ele comprar o colchão se ele não passar através das portas de sua casa. Portanto ele quer saber se consegue passar o colchão pelas portas e para isso precisa de sua ajuda.

Entrada

A primeira linha da entrada contém três números inteiros A, B e C, as três dimensões do colchão, em centímetros. A segunda linha contém dois inteiros H e L, respectivamente a altura e a largura das portas em centímetros.

Saída

Se programa deve escrever uma única linha, contendo apenas a letra 'S' se o colchão passa pelas portas e apenas a letra 'N' em caso contrário.

Restrições

- $1 \leq A, B, C \leq 300$
- $1 \leq H, L \leq 250$

Exemplos

Entrada	**Saída**
25 120 220 200 100	S

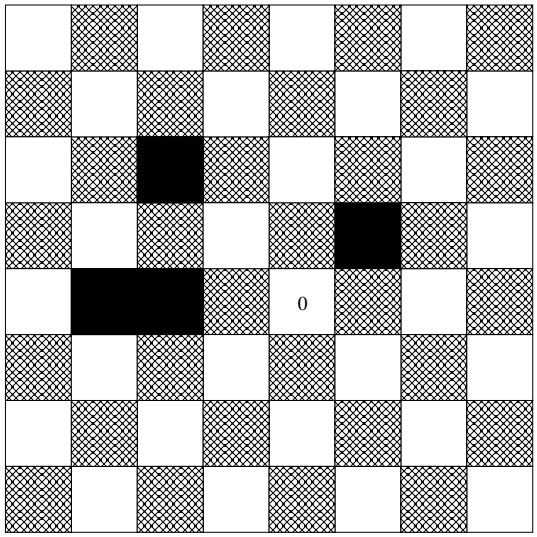
Entrada	**Saída**
25 205 220 200 100	N

Entrada	**Saída**
25 200 220 200 100	S

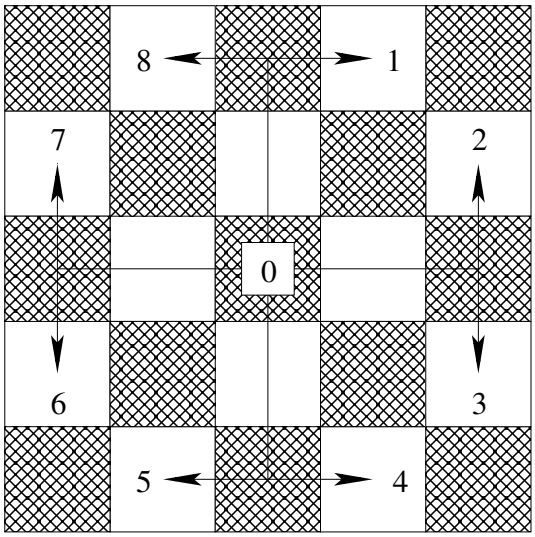
O Tabuleiro Esburacado

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

Um tabuleiro normal, 8 x 8, foi danificado, e 4 posições ficaram esburacadas. A Figura 1(a) mostra o tabuleiro. A posição inferior esquerda tem coordenadas $(0, 0)$. Os 4 buracos estão marcados em preto, e têm coordenadas $(1, 3)$, $(2, 3)$, $(2, 5)$ e $(5, 4)$. Um cavalo de xadrez foi colocado na posição $(4, 3)$, marcada como 0 no tabuleiro.



(a)



(b)

Os 8 movimentos de um cavalo estão numerados de 1 a 8 na Figura 1(b), a partir da posição marcada como 0. Por exemplo, se o cavalo estiver na posição inicial $(4, 3)$, o movimento 7 leva o cavalo à posição $(2, 4)$, sem cair no buraco $(2, 3)$, porque o cavalo salta da posição $(4, 3)$ para a posição $(2, 4)$.

Seu problema é simular um passeio do cavalo, dados os movimentos através dos números de 1 a 8 e determinar quantos movimentos o cavalo faz até ou (i) terminar o passeio ou (ii) cair em um buraco. Por exemplo, na trajetória dada pelos 5 movimentos $1, 8, 5, 3, 4$, o cavalo passa pelas posições $(5, 5)$, $(4, 7)$, $(3, 5)$ e cai no buraco $(5, 4)$, fazendo portanto apenas 4 movimentos.

Já no passeio dado pelos 3 movimentos $6, 8, 1$, o cavalo passa pelas posições $(2, 2)$, $(1, 4)$ e $(2, 6)$ e não cai em nenhum buraco: portanto, perfaz todos os 3 movimentos do passeio.

Entrada

A primeira linha da entrada contém N, o número de movimentos do passeio. A segunda linha contém N inteiros $M_1, M_2, \ldots, M_N$, separados por um espaço em branco, correspondentes aos N movimentos do cavalo no passeio. Um movimento pode levar o cavalo a cair em um buraco, mas nunca leva o cavalo a sair do tabuleiro.

Saída

Seu programa deve imprimir uma única linha, contendo um único número inteiro, o número de movimentos do cavalo até terminar o passeio ou o cavalo cair em um buraco.

Restrições

- $1 \leq N \leq 100$

- $1 \leq M_I \leq 8$, para $I = 1, 2, \ldots, N$.

Exemplos

Entrada	**Saída**
5 1 8 5 3 4	4

Entrada	**Saída**
3 6 8 1	3

Frequencia na aula

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

Certa vez, numa aula, a professora passou um filme para os alunos assistirem. Durante este filme, ela passou uma lista de presença em sua sala para verificar a presença dos alunos, onde cada aluno deveria inserir apenas seu número de registro. Alguns alunos contudo, como possuem amigos que fogem da aula, decidiram ser camaradas e inseriram os números de registro de seus amigos fujões. O problema é que muitos alunos são amigos de alunos que fogem da aula e alguns números de registro acabaram sendo repetidamente inseridos na lista de presença. Além de tudo, alguns dos alunos que se esperava que não estivessem na aula de fato estavam!

A professora, ao notar que a lista de presença continha alguns números repetidos, ficou sem entender, mas decidiu dar um voto de confiança e dar presença a todos os alunos cujos números de registro estavam na lista. Como são muitos alunos na sala e muitos números com repetição, ela pediu a sua ajuda para determinar o total de alunos que receberam presença na aula.

Entrada

A primeira linha da entrada contém um número inteiro N, que informa a quantidade de números de registro que apareceram na lista de presença. Cada uma das N linhas seguintes contém um número de registro V_i que foi inserido na lista de presença.

Saída

Seu programa deve imprimir uma única linha, contendo apenas um número inteiro, o número de alunos que receberam presença.

Restrições

- $1 \leq N \leq 10^5$
- Para cada elemento V_i, $0 \leq V_i \leq 10^6$

Informações sobre pontuação

- Em um conjunto de casos que totaliza 40 pontos, $N \leq 10^3$ e $V_i \leq 10^3$

Exemplos

Entrada	**Saída**
3 2 3 1	3

Entrada	**Saída**
7 0 5 12 41 7 5 41	5

Tarzan

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

Tarzan vive na floresta e é o responsável por manter a ordem na região onde vive. Para locomover-se entre as árvores ele só usa cipós pois esse é um meio de transporte muito mais rápido e seguro do que andar no chão da selva, além de, é claro, poder soltar seu grito característico enquanto viaja.

Os cipós das árvores têm todos o mesmo alcance. Dessa forma, é possível viajar de cipó de uma árvore para outra se a distância entre elas é no máximo D, onde D é o alcance dos cipós.

Recentemente uma forte chuva assolou a região e derrubou algumas árvores, restando na floresta apenas N árvores. Agora Tarzan quer saber se ele consegue viajar de cipó entre todas árvores remanescentes para poder continuar mantendo a ordem na região.

Para poder manter a ordem ele precisa ser capaz de, partindo de qualquer uma das árvores, poder chegar a todas as outras árvores remanescentes, possivelmente passando por outras árvores no caminho, sempre utilizando somente cipós.

Entrada

A primeira linha da entrada contém dois inteiros, N e D, indicando respectivamente o número de árvores remanescentes e o alcance dos cipós. Cada uma das N linhas seguintes contém dois inteiros X_i e Y_i, as coordenadas da i-ésima árvore. Não existem duas árvores com as mesmas coordenadas.

Saída

Seu programa deve escrever uma única linha, contendo um único caractere: '`S`' se Tarzan consegue viajar de cipó entre todas as árvores remanescentes, e '`N`' caso contrário.

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $N \leq 10$
- Em um conjunto de casos de teste que totaliza 70 pontos, $N \leq 100$

Restrições

- $2 \leq N \leq 1000$
- $1 \leq D \leq 5000$
- $0 \leq X_i, Y_i \leq 5000$

Exemplos

Entrada	**Saída**
4 5 1 1 6 1 6 6 11 6	S

Entrada	**Saída**
4 5 1 1 6 6 11 6 13 8	N