OBI2005

Caderno de Tarefas

Modalidade Programação • Nível 1

A PROVA TEM DURAÇÃO DE TRÊS HORAS

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 8 páginas (não contando esta folha de rosto), numeradas de 1 a 8. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar 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*. 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*.
- 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.

Sociedade Brasileira de Computação
www.sbc.org.br

Frota de Táxi

Iniciante - Fácil

Arquivo fonte: taxi.c, taxi.cc, taxi.cpp ou taxi.pas

A Companhia de Táxi Tabajara (CTT) é uma das maiores empresas de transporte do país. Possui uma vasta frota de carros e opera em todas as grandes cidades. Recentemente a CTT modernizou a sua frota, adquirindo um lote de 500 carros bi-combustíveis (carros que podem utilizar como combustível tanto álcool quanto gasolina). Além do maior conforto para os passageiros e o menor gasto com manutenção, com os novos carros é possível uma redução adicional de custo: como o preço da gasolina está sujeito a variações muito bruscas e pode ser vantagem, em certos momentos, utilizar álcool como combustível. Entretanto, os carros possuem um melhor desempenho utilizando gasolina, ou seja, em geral, um carro percorre mais quilômetros por litro de gasolina do que por litro de álcool.

Tarefa

Você deve escrever um programa que, dados o preço do litro de álcool, o preço do litro de gasolina e os quilômetros por litro que um carro bi-combustível realiza com cada um desses combustíveis, determine se é mais econômico abastecer os carros da CTT com álcool ou com gasolina. No caso de não haver diferença de custo entre abastecer com álcool ou gasolina a CTT prefere utilizar gasolina.

Entrada

A entrada é composta por uma linha contendo quatro números reais com precisão de duas casas decimais A, G, R_a e R_g, representando respectivamente o preço por litro do álcool, o preço por litro da gasolina, o rendimento (km/l) do carro utilizando álcool e o rendimento (km/l) do carro utilizando gasolina.

A entrada deve ser lida do dispositivo de entrada padrão (normalmente o teclado).

Saída

A saída deve ser composta por uma única linha contendo o caractere 'A' se é mais econômico abastecer a frota com álcool ou o caractere 'G' se é mais econômico ou indiferente abastecer a frota com gasolina.

A saída deve ser escrita no dispositivo de saída padrão (normalmente a tela).

Restrições

$0.01 \leq A \leq 10.00$
$0.01 \leq G \leq 10.00$
$0.01 \leq R_a \leq 20.00$
$0.01 \leq R_g \leq 20.00$

Exemplos de entrada e saída

Exemplo 1

Entrada	**Saída**
`1.20 2.30 10.00 15.00`	`A`

Exemplo 2

Entrada	**Saída**
`1.00 1.00 9.00 9.01`	`G`

Exemplo 3

Entrada	**Saída**
`1.00 1.00 11.00 11.00`	`G`

Campo de Minhocas

Iniciante - Médio

Arquivo fonte: minhoca.c, minhoca.cc, minhoca.cpp ou minhoca.pas

Minhocas são muito importantes para a agricultura e como insumo para produção de ração animal. A Organização para Bioengenharia de Minhocas (OBM) é uma entidade não governamental que promove o aumento da produção, utilização e exportação de minhocas.

Uma das atividades promovidas pela OBM é a manutenção de uma fazenda experimental para pesquisa de novas tecnologias de criação de minhocas. Na fazenda, a área destinada às pesquisas é de formato retangular, dividida em células quadrangulares de mesmo tamanho. As células são utilizadas para testar os efeitos, na produção de minhocas, de variações de espécies de minhocas, tipos de terra, de adubo, de tratamento, etc. Os pesquisadores da OBM mantêm um acompanhamento constante do desenvolvimento das minhocas em cada célula, e têm uma estimativa extremamente precisa da produtividade em cada uma das células. A figura abaixo mostra um mapa da fazenda, mostrando a produtividade estimada de cada uma das células.

81	28	240	10
40	10	100	240
20	180	110	35

Um pesquisador da OBM inventou e construiu uma máquina colhedeira de minhocas, e quer testá-la na fazenda. A máquina tem a largura de uma célula, e em uma passada pelo terreno de uma célula colhe todas as minhocas dessa célula, separando-as, limpando-as e empacotando-as. Ou seja, a máquina eliminará uma das etapas mais intensivas de mão de obra no processo de produção de minhocas. A máquina, porém, ainda está em desenvolvimento e tem uma restrição: não faz curvas, podendo movimentar-se somente em linha reta.

Decidiu-se então que seria efetuado um teste com a máquina, de forma a colher o maior número possível de minhocas em uma única passada, em linha reta, de lado a lado do campo de minhocas. Ou seja, a máquina deve colher todas as minhocas de uma 'coluna' ou de uma 'linha' de células do campo de minhocas (a linha ou coluna cuja soma das produtividades esperadas das células é a maior possível).

Tarefa

Escreva um programa que, fornecido o mapa do campo de minhocas, descrevendo a produtividade estimada em cada célula, calcule o número esperado total de minhocas a serem colhidas pela máquina durante o teste, conforme descrito acima.

Entrada

A primeira linha da entrada contém dois números inteiros N e M, representando respectivamente o número de linhas ($1 \leq N \leq 100$) e o número de colunas ($1 \leq M \leq 100$) de células existentes no campo experimental de minhocas. Cada uma das N linhas seguintes contém M inteiros, representando as produtividades estimadas das células correspondentes a uma linha do campo de minhocas.

A entrada deve ser lida do dispositivo de entrada padrão (normalmente o teclado).

Saída

A saída deve ser composta por uma única linha contendo um inteiro, indicando o número esperado total de minhocas a serem colhidas pela máquina durante o teste.

A saída deve ser escrita no dispositivo de saída padrão (normalmente a tela).

Restrições

$1 \leq N \leq 100$
$1 \leq M \leq 100$
$0 \leq$ *Produtividade de uma célula* ≤ 500
$0 \leq$ *Produtividade de uma linha ou coluna de células* ≤ 50000

Exemplos de entrada e saída

Exemplo 1

Entrada	**Saída**

`3 4`
`81 28 240 10`
`40 10 100 240`
`20 180 110 35`
	`450`

Exemplo 2

Entrada	**Saída**

`4 1`
`100`
`110`
`0`
`100`
	`310`

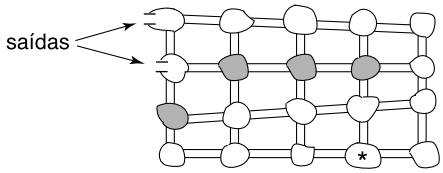
Duende Perdido

Gráfos - Fácil

Arquivo fonte: duende.c, duende.cc, duende.cpp ou duende.pas

Gugo, o duende, ficou preso em uma caverna e precisa sair o mais rapidamente possível. A caverna é formada por salões interligados por túneis, na forma de uma grade retangular, com N linhas e M colunas. Alguns dos salões da caverna têm paredes de cristal. Duendes, como todos sabem, não gostam de ficar em ambientes com qualquer tipo de cristal, pois seus organismos entram em ressonância com a estrutura de cristais, e em casos extremos os duendes podem até mesmo explodir. Compreensivelmente, Gugo não quer entrar em nenhum salão com parede de cristal.

A figura abaixo mostra uma caverna com quatro linhas e cinco colunas de salões; os salões cinza têm paredes de cristal. A posição inicial de Gugo é indicada com um caractere '*'.



Tarefa

Você deve escrever um programa que, dadas a configuração da caverna e a posição inicial de Gugo dentro da caverna, calcule qual o número mínimo de salões pelos quais o duende deve passar antes de sair da caverna (não contando o salão em que o duende está inicialmente), mas contando o salão que tem saída para o exterior).

Entrada

A caverna será modelada como uma matriz de duas dimensões, cujos elementos representam os salões. Um salão que não tem parede de cristal e que tem saída para o exterior da caverna é representado pelo valor 0; um salão que não tem parede de cristal e não tem saída para o exterior é representado pelo valor 1; um salão que tem parede de cristal é representado pelo valor 2; e o salão em que o duende está inicialmente (que não tem saída para o exterior e nem paredes de cristal) é representado pelo valor 3. A figura abaixo mostra a representação da caverna apresentada acima.

0	1	1	1	1
0	2	2	2	1
2	1	1	1	1
1	1	1	3	1

A primeira linha da entrada contém dois números inteiros N e M que indicam respectivamente o número de linhas ($1 \leq N \leq 10$) e o número de colunas ($1 \leq M \leq 10$) da representação da caverna. Cada uma das N linhas seguintes contém M números inteiros C_i, descrevendo os salões da caverna e a posição inicial do duende ($0 \leq C_i \leq 3$). Você pode supor que sempre há um trajeto que leva Gugo à saída da caverna.

A entrada deve ser lida do dispositivo de entrada padrão (normalmente o teclado).

Saída

Seu programa deve produzir uma única linha na saída, contendo um número inteiro representando a quantidade mínima de salões pelos quais Gugo deve passar antes de conseguir sair da caverna (não contando o salão em que ele está inicialmente, mas contando o salão que tem saída para o exterior).

A saída deve ser escrita no dispositivo de saída padrão (normalmente a tela).

Restrições

$1 \leq N \leq 10$
$1 \leq M \leq 10$
$0 \leq C_i \leq 3$

Exemplos de entrada e saída

Exemplo 1

Entrada	**Saída**
4 5 0 1 1 1 1 0 2 2 2 1 2 1 1 1 1 1 1 1 3 1	8

Exemplo 2

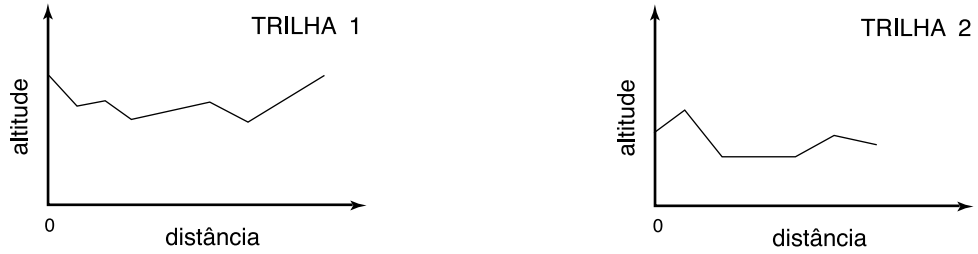
Entrada	**Saída**
1 10 2 0 1 1 3 1 1 1 0 1	3

Trilhas

Matemática - Fácil

Arquivo fonte: trilhas.c, trilhas.cc, trilhas.cpp ou trilhas.pas

Nos finais de semana Paulo faz longas caminhadas pelas bonitas trilhas que atravessam as matas vizinhas à sua cidade. Recentemente Paulo adquiriu um aparelho de GPS (siglas do inglês *Sistema de Posicionamento Global*) e com ele mapeou as mais belas trilhas da região. Paulo programou o GPS para armazenar, a intervalos regulares, a altitude do ponto corrente durante o trajeto. Assim, após percorrer as trilhas com o seu GPS, Paulo tem informações que permitem por exemplo produzir gráficos como os abaixo:



Paulo tem uma nova namorada, e quer convencê-la a passear junto com ele pelas trilhas. Para o primeiro passeio juntos, Paulo quer escolher uma trilha "fácil". Segundo o seu critério, a trilha mais fácil é a que, em um dos sentidos do percurso, exige o menor esforço de subida. O esforço exigido em um trecho de subida é proporcional ao desnível do trecho.

Tarefa

Dadas as informações colhidas por Paulo sobre distâncias e altitudes de um conjunto de trilhas, você deve escrever um programa que determine qual é a trilha que exige o menor esforço de subida.

Entrada

A primeira linha da entrada contém um número inteiro N que indica o número de trilhas. Cada uma das N linhas seguintes contém a descrição de uma trilha ($1 \leq N \leq 100$). As trilhas são identificadas por números de 1 a N. A ordem em que as trilhas aparecem na entrada determina os seus identificadores (a primeira trilha é a de número 1, a segunda a de número 2, a última a de número N). A descrição de uma trilha inicia com um número inteiro M que indica a quantidade de pontos de medição da trilha ($2 \leq M \leq 1000$), seguido de M números inteiros H_i representando a altura dos pontos da trilha (medidos a intervalos regulares e iguais para todas as linhas). Paulo pode percorrer a trilha em qualquer sentido (ou seja, partindo do ponto de altitude H_1 em direção ao ponto de altitude H_M, ou partindo do ponto de altitude H_M em direção ao ponto de altitude H_1).

A entrada deve ser lida do dispositivo de entrada padrão (normalmente o teclado).

Saída

Seu programa deve produzir uma única linha na saída, contendo um número inteiro representando o identificador da melhor trilha, conforme determinado pelo seu programa. Em caso de empate entre duas ou mais trilhas, imprima a de menor identificador.

A saída deve ser escrita no dispositivo de saída padrão (normalmente a tela).

Restrições

$1 \leq N \leq 100$
$2 \leq M \leq 1000$
$0 \leq H_i \leq 1000$

Exemplos de entrada e saída

Exemplo 1

Entrada	**Saída**
5 4 498 500 498 498 10 60 60 70 70 70 70 80 90 90 100 5 200 190 180 170 160 2 1000 900 4 20 20 20 20	2

Exemplo 2

Entrada	**Saída**
3 5 600 601 600 601 600 4 500 499 500 499 4 300 300 302 300	2