OBI2011

Caderno de Tarefas

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

26 de março de 2011

A PROVA TEM DURAÇÃO DE **3** HORAS

Promoção:

Sociedade Brasileira de Computação

Patrocínio:

Fundação Carlos Chagas

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

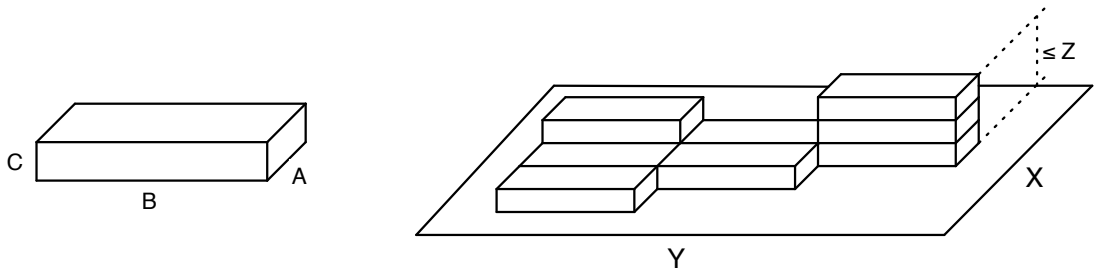
- Este caderno de tarefas é composto por 7 páginas (não contando a folha de rosto), numeradas de 1 a 7. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta 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.

Transporte de Contêineres

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

A Betalândia é um país que apenas recentemente se abriu para o comércio exterior e está preparando agora sua primeira grande exportação. A Sociedade Betalandesa de Comércio (SBC) ficou encarregada de conduzir a exportação e determinou que, seguindo os padrões internacionais, a carga será transportada em contêineres, que são, por sua vez, colocados em grandes navios para o transporte internacional.

Todos os contêineres betalandeses são idênticos, medindo A metros de largura, B metros de comprimento e C metros de altura. Um navio porta-contêineres pode ser visto como um retângulo horizontal de X metros de largura e Y metros de comprimento, sobre o qual os contêineres são colocados. Nenhuma parte de contêiner pode ficar para fora do navio. Além disso, para possibilitar a travessia de pontes, a altura máxima da carga no navio não pode ultrapassar Z metros.



Devido a limitações do guindaste utilizado, os contêineres só podem ser carregados alinhados com o navio. Ou seja, os contêineres só podem ser colocados sobre o navio de tal forma que a largura e o comprimento do contêiner estejam paralelos à largura e ao comprimento do navio, respectivamente.

A SBC está com problemas para saber qual a quantidade máxima de contêineres que podem ser colocados no navio e pede sua ajuda. Sua tarefa, neste problema, é determinar quantos contêineres podem ser carregados no navio respeitando as restrições acima.

Entrada

A entrada consiste de duas linhas. A primeira linha contém três inteiros A, B e C que representam as dimensões dos contêineres, enquanto a segunda linha contém outros três inteiros X, Y e Z que representam as dimensões do navio.

Saída

Seu programa deve imprimir apenas uma linha contendo um inteiro que indica a quantidade máxima de contêineres que o navio consegue transportar.

Restrições

- $1 \leq A, B, C, X, Y, Z \leq 10^6$
- É garantido que a maior resposta será menor ou igual a 10^6.

Exemplos

Entrada	**Saída**
1 1 1 1 1 1	1

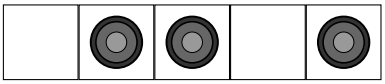
Entrada	**Saída**
1 2 5 9 6 11	54

Entrada	**Saída**
1 2 12 6 9 10	0

Campo Minado

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

Leonardo Viana é um garoto fascinado por jogos de tabuleiro. Nas férias de janeiro, ele aprendeu um jogo chamado "Campo minado", que é jogado em um tabuleiro com N células dispostas na horizontal. O objetivo desse jogo é determinar, para cada célula do tabuleiro, o número de minas explosivas nos arredores da mesma (que são a própria célula e as células imediatamente vizinhas à direita e à esquerda, caso essas existam). Por exemplo, a figura abaixo ilustra uma possível configuração de um tabuleiro com 5 células:



A primeira célula não possui nenhuma mina explosiva, mas é vizinha de uma célula que possui uma mina explosiva. Nos arredores da segunda célula temos duas minas, e o mesmo acontece para a terceira e quarta células; a quinta célula só tem uma mina explosiva em seus arredores. A próxima figura ilustra a resposta para esse caso.

Leonardo sabe que você participa da OBI e resolveu lhe pedir para escrever um programa de computador que, dado um tabuleiro, imprima o número de minas na vizinhança de cada posição. Assim, ele poderá conferir as centenas de tabuleiros que resolveu durante as férias.

Entrada

A primeira linha da entrada contém um inteiro N indicando o número de células no tabuleiro. O tabuleiro é dado nas próximas N linhas. A i-ésima linha seguinte contém 0 se não existe mina na i-ésima célula do tabuleiro e 1 se existe uma mina na i-ésima célula do tabuleiro.

Saída

A saída é composta por N linhas. A i-ésima linha da saída contém o número de minas explosivas nos arredores da i-ésima célula do tabuleiro.

Restrições

- $1 \leq N \leq 50$

Exemplos

Entrada	**Saída**
5 0 1 1 0 1	1 2 2 2 1

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

Corrida

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

A escola de Joãozinho tradicionalmente organiza uma corrida ao redor do prédio. Como todos os alunos são convidados a participar e eles estudam em períodos diferentes, é difícil que todos corram ao mesmo tempo.

Para contornar esse problema, os professores cronometram o tempo que cada aluno demora para dar cada volta ao redor da escola, e depois comparam os tempos para descobrir a classificação final.

Sua tarefa é, sabendo o número de competidores, o número de voltas de que consistiu a corrida e os tempos de cada aluno competidor, descobrir quem foi o aluno vencedor, para que ele possa receber uma medalha comemorativa.

Entrada

A primeira linha da entrada contém dois inteiros N e M representando o número de competidores e o número de voltas da corrida, respectivamente.

Cada uma das N linhas seguintes representa um competidor: a primeira linha representa o primeiro competidor, a segunda linha representa o segundo competidor, e assim por diante. Cada linha contém M inteiros representando os tempos em cada volta da corrida: o primeiro inteiro é o tempo da primeira volta, o segundo inteiro é o tempo da segunda volta, e assim por diante.

Garante-se que não houve dois competidores que gastaram o mesmo tempo para completar a corrida inteira.

Saída

A saída consiste de um único inteiro, que corresponde ao número do competidor que ganhou a corrida.

Restrições

- $2 \leq N \leq 100$
- $1 \leq M \leq 100$
- $1 \leq$ qualquer número da entrada que represente o tempo de uma volta $\leq 10^6$

Exemplos

Entrada	**Saída**
2 1 5 7	1

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

Neste exemplo, existem três competidores numa corrida de três voltas. Os tempos de cada competidor em cada volta foram como na tabela a seguir.

	Volta 1	Volta 2	Volta 3	Tempo total
Competidor 1	3	5	6	14
Competidor 2	1	2	3	6
Competidor 3	1	1	1	3

Sendo assim, o vencedor foi o competidor 3 (com um tempo total de 3).