OBI2015

Caderno de Tarefas

Modalidade **Programação** • **Nível 2** • Fase **2**

29 de agosto de 2015

A PROVA TEM DURAÇÃO DE **5** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

v1.2

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

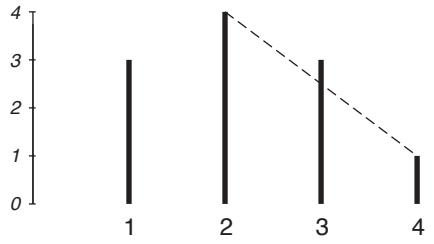
- Este caderno de tarefas é composto por 10 páginas (não contando a folha de rosto), numeradas de 1 a 10. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta 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; soluções na linguagem Python devem ser arquivos com sufixo *.py*; e soluções na linguagem Javascript devem ter arquivos com sufixo *.js*. 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 pen-drive, 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,input,print,write*
  - em Javascript: *scanf, printf*
- 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.

Macacos me mordam!

Nome do arquivo: `macacos.c`, `macacos.cpp`, `macacos.pas`, `macacos.java`, `macacos.js` *ou* `macacos.py`

Em uma floresta há N árvores alinhadas. A i-ésima árvore tem altura H_i e está localizada na posição X_i da floresta. Obi, o macaco camarada, está na primeira árvore da floresta, e deseja ir até a última árvore da floresta, porque ele ouviu dizer que há muitas bananas esperando por ele lá.

Para ir até a última árvore, Obi vai pular entre as árvores. Obi é um macaco muito ágil, e consegue pular de uma árvore A para outra árvore B sempre que, do topo da árvore A ele consegue enxergar o topo da árvore B, independente das posições das árvores A e B. Mas Obi é também um macaco muito preguiçoso, e quer pular o menor número de vezes possível.



Na figura acima podemos ver que, do topo da árvore na posicão 2, Obi não consegue enxergar o topo da árvore na posição 4, e portanto ele não pode pular de uma para outra sem passar pela árvore na posição 3. Assim, para o caso da figura acima, para ir da árvore 1 para a árvore 4 ele tem que passar por todas as árvores, dando um total de três pulos.

Dada a descrição da floresta, você deve escrever um programa para determinar o menor número de pulos que Obi deve dar para ir da primeira à última árvore da floresta.

Entrada

A primeira linha da entrada contém um número N, indicando a quantidade de árvores na floresta. Cada uma das N linhas seguintes descreve uma árvore da floresta, e contém dois inteiros X_i e H_i, respectivamente a posição e a altura de uma árvore. Cada árvore ocupa uma posição distinta na floresta (ou seja, não há duas árvores com o mesmo valor X_i).

Saída

Seu programa deve produzir uma única linha, contendo um único número inteiro, a menor a quantidade de pulos que Obi deve dar para ir da primeira até a última árvore da floresta.

Restrições

- $2 \leq N \leq 3 \times 10^5$
- $1 \leq H_i, X_i \leq 10^9$

Informações sobre a pontuação

- Em um conjunto de casos de teste cuja soma é 40 pontos: $2 \leq N \leq 300$

Exemplos

Entrada	**Saída**
4 1 3 2 4 3 3 4 1	3

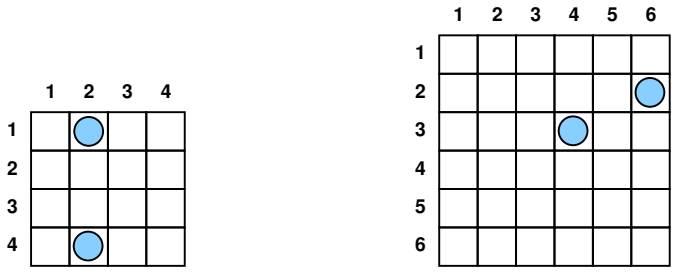
Entrada	**Saída**
4 1 3 2 4 3 3 4 2	2

Entrada	**Saída**
10 3 7 1 3 5 6 6 6 9 6 8 15 12 5 13 1 10 9 14 2	3

Chocolate em barra

Nome do arquivo: `chocolate.c`, `chocolate.cpp`, `chocolate.pas`, `chocolate.java`, `chocolate.js` *ou* `chocolate.py`

Vô Quico comprou uma barra de chocolate para suas duas netas Lúcia e Beatriz. A barra é composta de N linhas e N colunas de quadrados, onde N é sempre um número par. Em exatamente dois quadrados, que podem estar em qualquer posição na barra, há uma figurinha colada. Vô Quico gostaria de dar dois pedaços de tamanhos iguais, um para cada neta, cada pedaço contendo uma figurinha. Mais precisamente, ele gostaria de dividir a barra bem na metade, com um único corte vertical ou horizontal, deixando uma figurinha em cada pedaço.



A figura acima mostra dois exemplos. A barra da esquerda, com $N = 4$, vô Quico pode dividir na metade com um corte horizontal, e cada metade contém uma figurinha. Mas a barra da direita, com $N = 6$, ele não consegue dividir em dois pedaços iguais, separando as figurinhas, com um único corte horizontal ou vertical.

Dados N e as posições das duas figurinhas, seu programa deve dizer se é, ou não, possível dividir a barra em dois pedaços de tamanhos iguais, com um único corte horizontal ou vertical, deixando uma figurinha em cada pedaço.

Entrada

A primeira linha da entrada contém um inteiro N, representando as dimensões da barra (número de linhas e de colunas). A segunda linha contém dois inteiros X_1 e Y_1, representando as coordenadas da primeira figurinha. A terceira linha contém dois inteiros X_2 e Y_2, representando as coordenadas da segunda figurinha.

Saída

Seu programa deve imprimir apenas uma linha contendo um único caractere: "`S`", caso seja possível dividir a barra em pedaços iguais com um único corte horizontal ou vertical, separando as figurinhas, ou "`N`" caso não seja possível.

Restrições

- $2 \leq N \leq 1000$, N é sempre par;
- $1 \leq X_1, Y_1, X_2, Y_2 \leq N$.

Exemplos

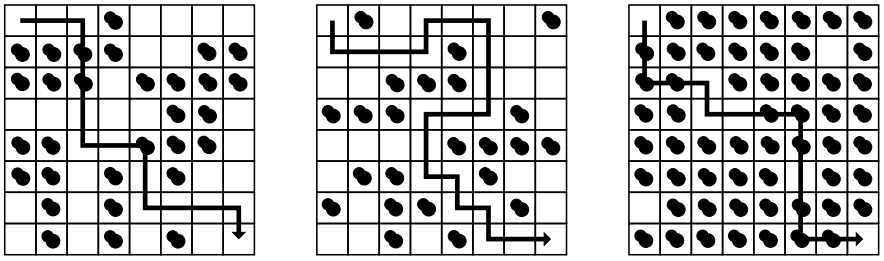
Entrada	**Saída**
4 1 2 4 2	S

Entrada	**Saída**
6 3 4 2 6	N

Mina

Nome do arquivo: `mina.c`, `mina.cpp`, `mina.pas`, `mina.java`, `mina.js` *ou* `mina.py`

Nossa mina de ouro será representada por N linhas e N colunas de quadrados. O mineiro está no quadrado inicial (superior esquerdo) e precisa cavar até o quadrado final (inferior direito), onde existe a maior concentração de ouro da mina. Alguns quadrados, porém, estão bloqueados por pedras, o que dificulta o trabalho. Sabendo que o mineiro pode realizar apenas movimentos ortogonais, seu programa deve calcular o número mínimo de quadrados bloqueados pelos quais o mineiro tem que passar para chegar no quadrado inferior direito. Os quadrados inicial e final nunca estão bloqueados. A figura abaixo ilustra três possíveis minas, para $N = 8$, para as quais os números mínimos de quadrados bloqueados são, respectivamente, três, zero e nove. A figura também mostra três possíveis trajetórias mínimas, como exemplo.



Entrada

A primeira linha da entrada contém um inteiro N, $2 \leq N \leq 100$, representando as dimensões da mina. Cada uma das N linhas seguintes contém N inteiros, definindo os quadrados da mina. O inteiro 0 representa um quadrado livre e o inteiro 1, um quadrado bloqueado.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o número mínimo de quadrados bloqueados pelos quais o mineiro tem que passar para chegar no quadrado final.

Exemplos

Entrada	**Saída**
6 0 1 0 0 0 0 1 1 0 0 1 1 1 0 1 1 1 1 0 0 0 1 1 0 0 0 1 1 1 0 0 1 0 0 0 0	3

Entrada	**Saída**
2 0 0 1 0	0

Cálculo

Nome do arquivo: `calculo.c`, `calculo.cpp`, `calculo.pas`, `calculo.java`, `calculo.js` *ou* `calculo.py`

Os computadores armazenam todas as informações usando representações binárias, ou seja, representações que utilizam apenas 0's e 1's. Há vários padrões para a representação de informação na forma binária, como por exemplo *"complemento-de-dois"* (usado para números inteiros), *"ascii"* (usado para caracteres e letras sem acentos), ou *"ieee-754"* (usado para números reais).

Neste problema vamos usar a representação *"obi-2015"* para certos valores positivos e menores do que 1. Na *"obi-2015"*, o número é representado por uma sequência de 0's e 1's de comprimento arbitrário. Lendo a representação da esquerda para a direita, o primeiro dígito binário representa o valor 2^{-1}, o segundo representa 2^{-2}, o terceiro 2^{-3}, e assim por diante. A representação utiliza sempre o menor número de dígitos possível (ou seja, desta forma o dígito mais à direita é sempre 1).

Por exemplo, a sequência de dígitos binários 0 1 representa o seguinte valor:

$$0 * 2^{-1} + 1 * 2^{-2} = 0.25$$

Já a sequência de dígitos binários 1 0 1 0 1 1 representa o seguinte valor:

$$1 * 2^{-1} + 0 * 2^{-2} + 1 * 2^{-3} + 0 * 2^{-4} + 1 * 2^{-5} + 1 * 2^{-6} = 0.671875$$

Sua tarefa é, dados dois números X e Y, representados no padrão *obi-2015*, determinar a representação da soma $X + Y$, também no padrão *obi-2015*.

Entrada

A primeira linha contém os inteiros M e N, representando respectivamente o número de dígitos binários de X e de Y. A segunda linha contém M números X_i, representando X no padrão *obi-2015*. A terceira linha contém N números Y_j, representando Y no padrão *obi-2015*.

Saída

Seu programa deve produzir uma única linha, contendo a representação do valor $X + Y$ no padrão *obi-2015*.

Restrições

- $1 \leq M, N \leq 10^3$
- $0 < X, Y < 1$
- $X_i \in \{0, 1\}$, para $0 \leq i \leq M$
- $Y_j \in \{0, 1\}$, para $0 \leq j \leq N$
- $X + Y < 1$

Informações sobre a pontuação

- Em um conjunto de casos de teste somando 20 pontos, $N \leq 5$ e $M \leq 5$.

Exemplos

Entrada	**Saída**
2 3 0 1 0 0 1	0 1 1

Entrada	**Saída**
5 4 1 0 1 1 1 0 0 0 1	1 1 0 0 1

Entrada	**Saída**
4 5 0 1 1 1 0 0 1 1 1	1 0 1 0 1

Fila

Nome do arquivo: `fila.c`, `fila.cpp`, `fila.pas`, `fila.java`, `fila.js` *ou* `fila.py`

Na cerimônia de encerramento da IOI, os competidores formam uma fila à medida que vão chegando ao local. Os competidores são desorganizados e entram na fila perto de seus novos amigos, ou seja, cada competidor escolhe uma posição arbitrária da fila para entrar. Logo na entrada do local há um telão que mostra fotografias e vídeos dos competidores durante a competição. Há uma grande diferença entre as alturas dos competidores, inclusive pelas diferenças de idade, e para que todos possam ver o telão, deve-se evitar que um competidor muito alto fique na frente de um competidor muito baixo, a não ser que esse competidor mais alto esteja longe, mais à frente na fila.

A organização da IOI está monitorando a fila e pediu que você faça um programa que inicialmente receba a descrição da fila inicial (número N de pessoas e suas alturas $A_1, A_2, \ldots, A_N$, pela ordem na fila, onde A_1 é a altura do primeiro da fila). Em seguida, seu programa deve processar dois tipos de operações:

- na operação tipo 0, seu programa recebe a informação que um novo competidor, de altura H, acabou de entrar na fila, exatamente atrás do I-ésimo competidor na fila (para $I = 0$ o novo competidor entrou no começo da fila)
- na operação tipo 1, seu programa recebe dois inteiros, I e D, e deve responder a uma consulta: considere a I-ésima pessoa na fila, digamos, P, e determine a posição na fila da pessoa mais próxima de P que está à frente de P e cuja altura é maior do que $H_I + D$ (onde H_I é a altura de P).

Entrada

A primeira linha da entrada contém um único número inteiro N, indicando o número de pessoas na fila inicial. A segunda linha da entrada contém os N números inteiros $A_1, A_2, \ldots, A_N$, as alturas de cada pessoa da fila. A terceira linha contém um único inteiro Q indicando o número de operações. Cada uma das Q linhas seguintes contém três números inteiros números T, I e X, descrevendo uma operação: T indica o tipo da operação, I representa uma posição na fila e X é a altura H do novo competidor (na operação tipo 0) ou o parâmetro D (na operação do tipo 1).

Restrições

- $0 \leq N \leq 6 \times 10^5$
- $1 \leq Q \leq 6 \times 10^5$
- $1 \leq A_i \leq 10^9$ para todo i, $1 \leq i \leq N$
- $1 \leq X \leq 10^9$

Informações sobre a pontuação

- Em um conjunto de casos de testes somando 40 pontos, $N \leq 2 \times 10^5$, $Q \leq 2 \times 10^5$ e todas as operações são do tipo 1;
- Em um conjunto de casos de testes somando 80 pontos, $N \leq 2 \times 10^5$ e $Q \leq 2 \times 10^5$.

Exemplos

Entrada	**Saída**
5 10 5 7 8 2 6 1 5 6 0 1 11 1 6 6 0 0 13 1 6 4 1 6 5	1 2 1 0