Competidor(a): _____

Número de inscrição: _____–_____ *(opcional)*

OBI2018

Caderno de Tarefas

Modalidade **Programação** • **Nível Sênior** • Fase **Estadual**

21 de junho de 2018

A PROVA TEM DURAÇÃO DE **2** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

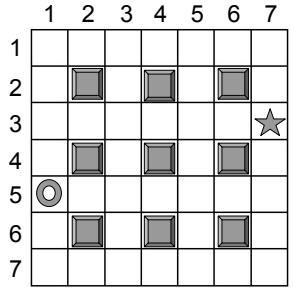
- Este caderno de tarefas é composto por 7 páginas (não contando a folha de rosto), numeradas de 1 a 7. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta 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; em particular, seu programa não deve escrever frases como "Digite o dado de entrada:" ou similares.
- 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 necessariamente 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 2 devem ser arquivos com sufixo *.py2*; soluções na linguagem Python 3 devem ser arquivos com sufixo *.py3*; e soluções na linguagem Javascript devem ter arquivos com sufixo *.js*.
- Na linguagem Java, **não** use o comando *package*, e note que o nome de sua classe principal deve usar somente letras minúsculas (o mesmo nome do arquivo indicado nas tarefas).
- Para tarefas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada tarefa.
- 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 a tarefa 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.

Fuga

Nome do arquivo: "`fuga.x`", *onde* `x` *deve ser* `c|cpp|pas|java|js|py2|py3`

Os irmãos Violet e Klaus estão fugindo pelas suas vidas do Conde Olaf, que corre atrás deles dentro de um prédio abandonado. Violet e Klaus acabam de entrar em uma sala retangular de largura N e comprimento M, dividida em $N \cdot M$ células (i, j) de área 1 ($1 \leq i \leq N$ e $1 \leq j \leq M$). Em algumas células dessa sala, existem armários. Toda célula (i, j) onde i e j são pares contém um armário. A sala tem uma entrada na célula (X_e, Y_e) e uma saída na célula (X_s, Y_s), que ficam em posições diferentes **nas bordas** da sala. A entrada e a saída nunca são adjacentes a um armário.

A figura a seguir mostra a uma possível configuração da sala, onde $N = M = 7$, a entrada fica na posição $(3, 7)$ (marcada com uma estrela) e a saída fica na posição $(5, 1)$ (marcada com um círculo). Os armários estão indicados em quadrados cinzas.



Para atrasar Conde Olaf, que os está perseguindo e entrará na sala em alguns momentos, os irmãos decidiram derrubar armários da sala, de forma a aumentar o tamanho do percurso necessário para ir da entrada até a saída. As células ocupadas por armários caídos ou em pé não podem ser percorridas. Um armário pode ser derrubado em qualquer uma das direções paralelas aos lados da sala e ocupa duas células após cair. Ou seja, um armário na posição (i, j) da sala, ao cair irá ocupar uma das seguintes opções:

- As células (i, j) e $(i, j + 1)$;
- As células (i, j) e $(i, j - 1)$;
- As células (i, j) e $(i + 1, j)$; ou
- As células (i, j) e $(i - 1, j)$.

Dadas as dimensões da sala e as posições de entrada e de saída, você deve encontrar uma forma de derrubar os armários tal que a distância entre a entrada e a saída da sala seja a maior possível dentre todas as formas de derrubar os armários.

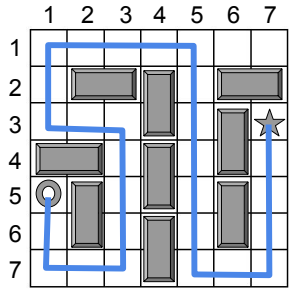
Para o exemplo acima, a figura abaixo é uma solução possível. Os retângulos cinzas representam os armários derrubados e a linha representa o caminho entre a entrada e a saída (que passa por 29 células). Nesse caso, não é possível derrubar os armários de forma que a distância entre a entrada e a saída seja maior que 29.

Entrada

A primeira linha contém dois inteiros N e M, a largura e o comprimento da sala, respectivamente. A segunda linha contém dois inteiros X_e e Y_e, identificando a célula de entrada da sala (X_e, Y_e). A terceira linha contém dois inteiros X_s e Y_s, identificando a célula de saída da sala (X_s, Y_s).

Saída

Seu programa deve produzir um inteiro representando o tamanho do menor caminho (em número de células) da entrada até a saída da sala após derrubar os armários de forma ótima.



Restrições

- $3 \leq N, M \leq 11$
- $3 \leq X_e, X_s \leq N$
- $3 \leq Y_e, Y_s \leq M$
- N, M, X_e, X_s, Y_e, Y_s são ímpares.

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 40 pontos, $1 \leq N, M \leq 7$.

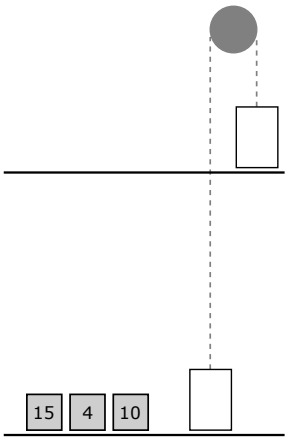
Exemplo de entrada 1	**Exemplo de saída 1**
7 7 3 7 5 1	29

Exemplo de entrada 2	**Exemplo de saída 2**
11 11 11 1 1 11	69

Elevador

Nome do arquivo: "`elevador.x`", *onde* `x` *deve ser* `c|cpp|pas|java|js|py2|py3`

Uma fábrica instalou um elevador composto de duas cabines ligadas por uma roldana, como na figura. Quando uma cabine sobe, a outra desce. No primeiro andar da fábrica existem algumas caixas de pesos diversos e precisamos levar todas as caixas para o segundo andar, usando o elevador. Apenas uma caixa pode ser colocada por vez dentro de uma cabine. Além disso, existe uma restrição de segurança importante: durante uma viagem do elevador, a diferença de peso entre as cabines pode ser no máximo de 8 unidades. De forma mais rigorosa, $P - Q \leq 8$, onde P é o peso da cabine mais pesada e Q, o peso da cabine mais leve. O gerente da fábrica não está preocupado com o número de viagens que o elevador vai fazer. Ele apenas precisa saber se é possível ou não levar todas as caixas para o segundo andar. No exemplo da figura, podemos levar todas as três caixas usando a seguinte sequência de seis viagens do elevador:



1. Sobe a caixa de peso 4, desce a outra cabine vazia; *(diferença de 4)*
2. Sobe a caixa de peso 10, desce a caixa de peso 4; *(diferença de 6)*
3. Sobe a caixa de peso 15, desce a caixa de peso 10; *(diferença de 5)*
4. Sobe a caixa de peso 4, desce a outra cabine vazia; *(diferença de 4)*
5. Sobe a caixa de peso 10, desce a caixa de peso 4; *(diferença de 6)*
6. Sobe a caixa de peso 4, desce a outra cabine vazia. *(diferença de 4)*

Dados os pesos de N caixas no primeiro andar, seu programa deve dizer se é possível ou não levar todas as N caixas para o segundo andar.

Entrada

A primeira linha da entrada contém um inteiro N indicando o número de caixas. A segunda linha da entrada contém N inteiros representando os pesos das caixas.

Saída

Imprima uma linha na saída. A linha deve conter o caracter `S` caso seja possível, ou `N` caso não seja possível levar todas as caixas até o segundo andar da fábrica.

Restrições

- $1 \leq N \leq 10^4$
- O peso das caixas está entre 1 e 10^5, inclusive.

Exemplo de entrada 1	**Exemplo de saída 1**
`3` `15 4 10`	`S`

Exemplo de entrada 2	**Exemplo de saída 2**
`8` `25 2 6 15 40 35 35 20`	`N`

Exemplo de entrada 3	**Exemplo de saída 3**
4 14 10 23 20	N

Exemplo de entrada 4	**Exemplo de saída 4**
1 8	S

Sequência

Nome do arquivo: "`sequencia.x`", *onde* `x` *deve ser* `c|cpp|pas|java|js|py2|py3`

O professor da importante disciplina de Indução Matemática está tentando resolver uma versão generalizada de um problema muito tradicional: encontrar o valor máximo possível para a soma dos elementos de uma subsequência contígua de uma sequência de números inteiros quaisquer. Mais rigorosamente, dado uma sequência $S = [s_1, s_2, \ldots, s_N]$, onde s_i é um número inteiro qualquer, para $1 \leq i \leq N$, maximizar $soma(i, j) = s_i + s_{i+1} + \cdots + s_j$ entre todos os possíveis pares (i, j), onde $1 \leq i \leq j \leq N$.

Na versão do professor, entretanto, alguns elementos da sequência são especiais e estão marcados. Além da sequência marcada, são dadas como entrada duas cotas: L e H, com $L \leq H$. O objetivo agora é encontrar o valor máximo possível para a soma dos elementos de uma subsequência contígua, que contenha pelo menos L e no máximo H elementos marcados.

Por definição, uma subsequência vazia (de zero elementos) tem soma igual a zero. Mas note que, como podemos ter uma cota inferior para o número de elementos marcados, a subsequência contígua de soma máxima pode ter soma negativa!

Entrada

A primeira linha da entrada contém três inteiros N, L e H, indicando respectivamente o número de elementos na sequência, a cota inferior L e a cota superior H. A segunda linha contém N inteiros s_i, para $1 \leq i \leq N$, definindo os elementos da sequência. A terceira linha contém N inteiros m_i, para $1 \leq i \leq N$, indicando as marcas. Se o i-ésimo elemento está marcado, o valor é $m_i = 1$. Se não estiver marcado, $m_i = 0$.

Saída

Imprima um inteiro, representando o valor máximo possível para a soma dos elementos de uma subsequência contígua, que contenha pelo menos L e no máximo H elementos marcados.

Restrições

- $1 \leq N \leq 10^5$
- $0 \leq L \leq H \leq 20$
- $-10^3 \leq s_i \leq 10^3$, para $1 \leq i \leq N$
- O número de elementos marcados na sequência é maior ou igual a L; portanto sempre existe solução.

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 15 pontos, $N \leq 10^2$
- Para um conjunto de casos de testes valendo 30 pontos, $N \leq 10^4$

Exemplo de entrada 1	**Exemplo de saída 1**

`14 3 4`
`9 0 -23 -12 7 1 -13 2 -1 9 -16 -1 14 12`
`1 0 0 1 0 1 0 0 1 1 0 0 1 1`
	`19`

Exemplo de entrada 2	**Exemplo de saída 2**
14 7 20 9 0 -23 -12 7 1 -13 2 -1 9 -16 -1 14 12 1 0 0 1 0 1 0 0 1 1 0 0 1 1	-12

Exemplo de entrada 3	**Exemplo de saída 3**
14 5 5 9 0 -23 -12 7 1 -13 2 -1 9 -16 -1 14 12 1 0 0 1 0 1 0 0 1 1 0 0 1 1	14

Exemplo de entrada 4	**Exemplo de saída 4**
14 0 20 9 0 -23 -12 7 1 -13 2 -1 9 -16 -1 14 12 1 0 0 1 0 1 0 0 1 1 0 0 1 1	26