OBI2014

Caderno de Tarefas

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

24 de maio de 2014

A PROVA TEM DURAÇÃO DE **3** HORAS

Promoção:

Sociedade Brasileira de Computação

Patrocínio:

Fundação Carlos Chagas

Instruções

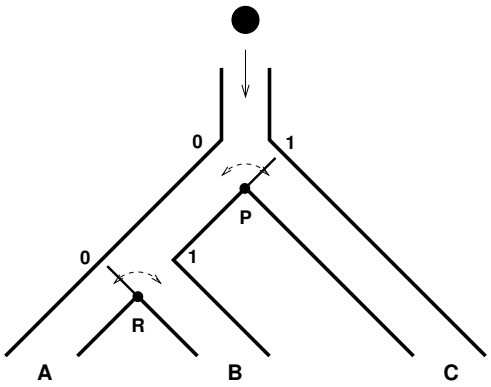
LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 4 páginas (não contando a folha de rosto), numeradas de 1 a 4. 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.

Flíper

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

Flíper é um tipo de jogo onde uma bolinha de metal cai por um labirinto de caminhos até chegar na parte de baixo do labirinto. A quantidade de pontos que o jogador ganha depende do caminho que a bolinha seguir. O jogador pode controlar o percurso da bolinha mudando a posição de algumas portinhas do labirinto. Cada portinha pode estar na posição 0, que significa virada para a esquerda, ou na posição 1 que quer dizer virada para a direita. Considere o flíper da figura abaixo, que tem duas portinhas. A portinha P está na posição 1 e a portinha R, na posição 0. Desse jeito, a bolinha vai cair pelo caminho B.



Você deve escrever um programa que, dadas as posições das portinhas P e R, neste flíper da figura, diga por qual dos três caminhos, A, B ou C, a bolinha vai cair!

Entrada

A entrada é composta por apenas uma linha contendo dois números P e R, indicando as posições das duas portinhas do flíper da figura.

Saída

A saída do seu programa deve ser também apenas uma linha, contendo uma letra maiúscula que indica o caminho por onde a bolinha vai cair: '`A`', '`B`' ou '`C`'.

Restrições

- O número P pode ser 0 ou 1. O número R pode ser 0 ou 1.

Exemplos

Entrada	**Saída**
`1 0`	`B`

Entrada	**Saída**
`0 0`	`C`

Gangorra

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

Joãozinho acaba de mudar de escola e a primeira coisa que percebeu na nova escola é que a gangorra do parquinho não é simétrica, uma das extremidades é mais longa que a outra. Após brincar algumas vezes com um amigo de mesmo peso, ele percebeu que quando está em uma extremidade, a gangorra se desequilibra para o lado dele (ou seja, ele fica na parte de baixo, e o amigo na parte de cima), mas quando eles trocam de lado, a gangorra se desequilibra para o lado do amigo. Sem entender a situação, Joãozinho pediu ajuda a outro amigo de outra série, que explicou que o comprimento do lado interfere no equilíbrio da gangorra, pois a gangorra estará equilibrada quando

$$P_1 * C_1 = P_2 * C_2$$

onde P_1 e P_2 são os pesos da criança no lado esquerdo e direito, respectivamente, e C_1 e C_2 são os comprimentos da gangorra do lado esquerdo e direito, respectivamente.

Com a equação, Joãozinho já consegue dizer se a gangorra está equilibrada ou não mas, além disso, ele quer saber para qual lado a gangorra descerá caso esteja desequilibrada.

Entrada

A primeira e única linha da entrada contém 4 inteiros, P_1, C_1, P_2 e C_2, nesta ordem.

Saída

Se a gangorra estiver equilibrada, imprima '`0`'. Se ela estiver desequilibrada de modo que a criança esquerda esteja na parte de baixo, imprima '`-1`', senão, imprima '`1`'.

Restrições

- $10 \leq P_1 \leq 100$ e $10 \leq C_1 \leq 100$
- $10 \leq P_2 \leq 100$ e $10 \leq C_2 \leq 100$

Exemplos

Entrada	**Saída**
`30 100 60 50`	`0`

Entrada	**Saída**
`40 40 38 60`	`1`

Entrada	**Saída**
`35 80 35 75`	`-1`

Fila

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

Com a proximidade da Copa do Mundo, o fluxo de pessoas nas filas para compra de ingressos aumentou consideravelmente. Como as filas estão cada vez maiores, pessoas menos pacientes tendem a desistir da compra de ingressos e acabam deixando as filas, liberando assim vaga para outras pessoas. Quando uma pessoa deixa a fila, todas as pessoas que estavam atrás dela dão um passo a frente, sendo assim nunca existe um espaço vago entre duas pessoas. A fila inicialmente contém N pessoas, cada uma com um identificador diferente. Joãozinho sabe o estado inicial dela e os identificadores em ordem das pessoas que deixaram a fila. Sabendo que após o estado inicial nenhuma pessoa entrou mais na fila, Joãozinho deseja saber o estado final da fila.

Entrada

A primeira linha contém um inteiro N representando a quantidade de pessoas inicialmente na fila. A segunda linha contém N inteiros representando os identificadores das pessoas na fila. O primeiro identificador corresponde ao identificador da primeira pessoa na fila. É garantido que duas pessoas diferentes não possuem o mesmo identificador. A terceira linha contém um inteiro M representando a quantidade de pessoas que deixaram a fila. A quarta linha contém M inteiros representando os identificadores das pessoas que deixaram a fila, na ordem em que elas saíram. É garantido que um mesmo identificador não aparece duas vezes nessa lista.

Saída

Seu programa deve imprimir uma linha contedo $N - M$ inteiros com os identificadores das pessoas que permaneceram na fila, em ordem de chegada.

Restrições

- $1 \leq N \leq 50000$
- $1 \leq M \leq 50000$ e $M < N$
- Cada identificador está entre 1 e 100000

Informações sobre a pontuação

- Em um conjunto de casos de teste equivalente a 30 pontos, $N \leq 1000$ e $M \leq 1000$.

Exemplos

Entrada	**Saída**
8 5 100 9 81 70 33 2 1000 3 9 33 5	100 81 70 2 1000

Entrada	**Saída**
4 10 9 6 3 1 3	10 9 6