OBI2009

Caderno de Tarefas

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

A PROVA TEM DURAÇÃO DE **5** HORAS

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 10 páginas (não contando esta 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*. 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.

Promoção:

Sociedade Brasileira de Computação

Patrocínio:

Fundação Carlos Chagas

Olimpíadas

Nome do arquivo fonte: `olimp.c`, `olimp.cpp`, *ou* `olimp.pas`

O Comitê Olímpico Internacional (COI) está visitando as cidades candidatas a sediar as Olimpíadas de 2016. O Rio de Janeiro é uma das cidades concorrentes, mas a competição é muito acirrada.

O COI tem um conjunto de exigências que devem ser obedecidas pelas cidades candidatas, como boas arenas para os jogos (ginásios, campos de futebol, pistas de atletismo, parque aquático,...), bons alojamentos, um plano para o tráfego de veículos durante os jogos, etc. Durante sua visita ao Rio de Janeiro, o COI colocou ainda mais uma exigência: a demonstração da qualidade dos sistemas de informática. Especificamente, o COI quer que a organização local demonstre a sua capacidade em informática produzindo um programa que gere a classificação final dos países, considerando o número de medalhas recebidas pelos atletas de cada país.

Tarefa

Sua tarefa é escrever um programa que, dada a informação dos países que receberam medalhas de ouro, prata e bronze em cada modalidade, gere a lista de classificação dos países na competição. Nesta tarefa, os países serão identificados por números inteiros. O melhor colocado deve ser o país que conseguiu o maior número de medalhas de ouro. Se houver empate entre países no número de medalhas de ouro, o melhor colocado entre esses é o país que conseguiu o maior número de medalhas de prata. Se houver empate também no número de medalhas de prata, o melhor colocado entre esses é o país que recebeu o maior número de medalhas de bronze. Se ainda assim houver empate entre dois países, o melhor classificado é o que tem o menor número de identificação.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado).

A primeira linha da entrada contém dois números inteiros N e M, separados por um espaço em branco, indicando respectivamente o número de países ($1 \leq N \leq 100$) e número de modalidades esportivas envolvidas na competição ($1 \leq M \leq 100$). Os países são identificados por números inteiros de 1 a N.

Cada uma das M linhas seguintes contém três números inteiros O, P e B, separados por um espaço em branco, representando os países cujos atletas receberam respectivamente medalhas de ouro ($1 \leq O \leq N$), prata ($1 \leq P \leq N$) e bronze ($1 \leq B \leq N$). Assim, se uma das M linhas contém os números `3 2 1`, significa que nessa modalidade a medalha de outro foi ganha pelo país `3`, a de prata pelo país `2` e a de bronze pelo país `1`.

Saída

Seu programa deve imprimir, na *saída padrão*, uma linha contendo N números, separados por um espaço em branco, representando os países na ordem decrescente de classificação (o primeiro número representa o país que é o primeiro colocado, o segundo número representa o país que é o segundo colocado, e assim por diante).

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $N \leq 10$ e $M \leq 10$.
- Em um conjunto de casos de teste que totaliza 70 pontos, $N \leq 50$ e $M \leq 50$.

Exemplos

Entrada	**Saída**
2 2 2 1 2 1 2 2	2 1

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

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

Simulador

Nome do arquivo fonte: `simulador.c`, `simulador.cpp`, *ou* `simulador.pas`

Um novo processador, denominado Faíska, está sendo desenvolvido para a empresa SBC. Este novo processador tem apenas duas instruções: *inversão* e *soma*, descritas a seguir.

- *Inversão*: dados dois endereços de memória X e Y, a operação `inverte(X,Y)` inverte a posição de palavras da memória de forma que
  - a palavra no endereço X troca de posição com a palavra de memória da posição Y;
  - a palavra no endereço $X + 1$ troca de posição com a palavra de memória da posição $Y - 1$;
  - a palavra no endereço $X + 2$ troca de posição com a palavra de memória da posição $Y - 2$;
  - e assim por diante, até que $X \geq Y$.
- *Soma*: dados dois endereços de memória X e Y, a operação `soma(X,Y)` imprime a soma das palavras de memória entre os endereços X e Y (inclusive).

Por exemplo, se a memória contém inicialmente, a partir da primeira posição de memória (endereço igual a 1) os valores `[1,2,3,4,5,6,7,8]`, a operação `inverte(3,7)` deixa a memória igual a `[1,2,7,6,5,4,3,8]`. Então, nesse estado, a execução de `soma(1,3)` produz a saída `10`.

Tarefa

Sua tarefa é escever um programa que, dada uma sequência de instruções do Faíska, simule a execução e produza o mesmo resultado que o Faíska produziria.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado).

A primeira linha da entrada contém dois números inteiros N e M, representando respectivamente o número palavras na memória ($1 \leq N \leq 10^9$) e o número de instruções do programa ($1 \leq M \leq 3000$). Cada uma das M linhas seguintes contém uma instrução do Faíska. Cada instrução é composta de um caratere descrevendo a instrução (‘`I`’ para inversão e ‘`S`’ para soma), seguido de um espaço, seguido de dois inteiros indicando os argumentos da instrução.

Inicialmente a configuração da memória é tal que cada palavra tem como conteúdo o seu próprio endereço. Em outras palavras, o conteúdo inicial da memória é `[1,2,3,...,N]`. Há pelo menos uma instrução `soma` em cada caso de teste.

Saída

Seu programa deve imprimir, na *saída padrão*, uma sequência de números inteiros, um em cada linha, indicando a saída gerada pelo Faíska.

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $N \leq 100$ e $M \leq 100$.
- Em um conjunto de casos de teste que totaliza 70 pontos, $N \leq 10^4$ e $M \leq 10^3$.

Exemplos

Entrada	**Saída**
10 2 I 1 5 S 3 7	19

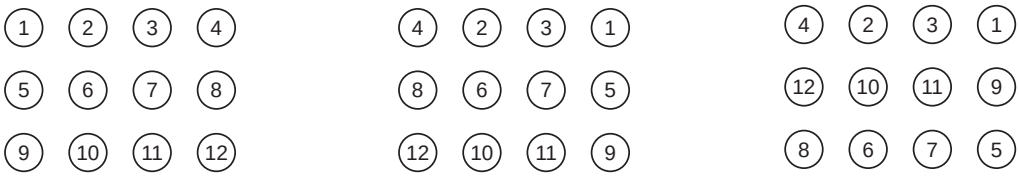
Entrada	**Saída**
15 4 S 2 11 I 10 15 I 1 10 S 5 10	65 21

Cadeiras do auditório

Nome do arquivo fonte: `cadeiras.c`, `cadeiras.cpp`, *ou* `cadeiras.pas`

As cadeiras do auditório da escola são organizadas em um quadriculado com L linhas e C colunas. As linhas são numeradas de 1 a L, as colunas são numeradas de 1 a C, e as cadeiras são numeradas de 1 a $L \times C$, de tal modo que uma cadeira na linha i coluna j tem o número $(i-1) \times C + j$.

Durante a aula de teatro, a professora fez com que os alunos executassem uma sequência de mudanças na configuração da sala. Cada uma dessas mudanças intercambiou ou duas colunas ou duas linhas. A figura abaixo ilustra uma configuração original com três linhas e quatro colunas, a posição das cadeiras após uma mudança (intercâmbio das colunas 1 e 4), e a posição das cadeiras após mais uma mudança (intercâmbio das linhas 2 e 3).



Configuração inicial das cadeiras

Após intercâmbio das colunas 1 e 4

Após intercâmbio das linhas 2 e 3

Ao final da aula, como era de se esperar, a numeração das cadeiras ficou bem bagunçada. O problema é que a próxima aula é de Matemática, e o professor é muito exigente, e quer começar a aula com as cadeiras perfeitamente posicionadas da maneira original.

Tarefa

Sua tarefa é escrever um programa que, dada a posição de cada cadeira ao final da aula de teatro, determine qual é a menor sequência de mudanças que devem ser executadas para retornar as cadeiras aos seus devidos lugares, considerando que cada mudança faça o intercâmbio ou de duas linhas ou de duas colunas de cadeiras.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado).

A primeira linha da entrada contém dois números inteiros L e C, representando respectivamente o número de linhas e o número de colunas de cadeiras do auditório ($1 \leq L \leq 200$ e $1 \leq C \leq 200$). Cada uma das L linhas seguintes contém C números inteiros entre 1 e $L \times C$, separados por um espaço em branco, indicando a posição das cadeiras ao final da aula de teatro. O j-ésimo número dado na linha i é o número da cadeira que se encontra na linha i e coluna j.

Saída

Seu programa deve imprimir, na *saída padrão*, na primeira linha um inteiro K representando o número de mudanças necessárias para retornar as cadeiras para sua posição original. Cada uma das K linhas seguintes contém a descrição de uma mudança, na forma de um caractere M (que pode ser '`L`' ou '`C`'), seguido de um espaço em branco, seguido de um inteiro X, seguido de um espaço em branco, seguido de um inteiro Y. Se o caractere descrevendo a mudança é '`L`', X e Y representam linhas que devem ser intercambiadas; se o caractere descrevendo a mudança é '`C`', X e Y representam colunas que devem ser intercambiadas.

Para todos os casos testes existe solução com $K \leq 1000$. Se mais de uma solução existe com o mesmo número de mudanças, imprima qualquer uma delas.

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $L \leq 10$ e $C \leq 10$.
- Em um conjunto de casos de teste que totaliza 70 pontos, $L \leq 100$ e $C \leq 100$.

Exemplos

Entrada	**Saída**

`2 2`
`4 3`
`2 1`
	`2`
	`L 1 2`
	`C 1 2`

Entrada	**Saída**

`3 4`
`1 2 3 4`
`5 6 7 8`
`9 10 11 12`
	`0`

Banda

Nome do arquivo fonte: `banda.c`, `banda.cpp`, *ou* `banda.pas`

Jimmy é um garoto muito esperto que adora música. No último mês ele ganhou um campeonato de um jogo cujo objetivo é tocar guitarra. Empolgado, Jimmy decidiu montar uma banda. Para Jimmy a banda perfeita tem quatro integrantes, ele e mais três: um baterista, um baixista e um cantor.

Agora Jimmy precisa encontrar os outros integrantes da banda. Para isto ele reuniu todos os álbums que encontrou na internet e, após escutá-los diversas vezes, compilou o que ele chama de *lista de entrosamento entre músicos*. Nessa lista ele atribui, para cada par de músicos que já tocaram juntos, uma nota inteira de 1 a 100, que é uma medida de quão bem os músicos tocam juntos (o *nível de entrosamento* entre eles). Se dois músicos nunca tocaram juntos o nível de entrosamento é zero. Jimmy nunca tocou com nenhum músico da lista.

Jimmy pretende formar a sua banda a partir da lista de entrosamento entre músicos, da seguinte maneira: ele quer escolher os outros três músicos de tal forma que a soma dos níveis de entrosamento dos integrantes da banda seja a maior possível (ou seja, a soma dos níveis de entrosamento dos três pares possíveis de serem formados entre os três novos integrantes seja a maior possível).

Mas a lista de entrosamento entre músicos ficou muito grande e Jimmy não está conseguindo escolher os integrantes. Por isso, Jimmy está pedindo sua ajuda.

Tarefa

Você deve ajudar Jimmy a montar a melhor banda possível fazendo um programa que receba uma lista contendo o nível de entrosamento para cada par de músicos que já tocaram junto, e determine os músicos que formariam a melhor banda.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado).

A primeira linha da entrada é formada por dois inteiros N e M, informando respectivamente o número de músicos ($3 \leq N \leq 100$) e o número de pares de músicos que já tocaram juntos ($0 \leq M \leq 10^4$). Os músicos são identificados por números inteiros de 1 a N. Cada uma das M linhas seguintes contém três inteiros X, Y e Z, em que X e Y representa um par de músicos ($1 \leq X \leq N$, $1 \leq Y \leq N$ e $X \neq Y$) e Z representa o seu nível de entrosamento ($1 \leq Z \leq 100$). Cada par de músicos que já tocou junto aparece uma única vez na entrada.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo três números inteiros separados por espaço em branco, identificando os três outros músicos que devem compor a banda (em qualquer ordem). Se existir mais de uma melhor banda, Jimmy contenta-se com qualquer uma.

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $N \leq 10$ e $M \leq 100$.
- Em um conjunto de casos de teste que totaliza 80 pontos, $N \leq 50$ e $M \leq 2450$.

Exemplos

Entrada	**Saída**
3 3 1 2 50 2 3 27 3 1 1	1 2 3

Entrada	**Saída**
5 8 1 2 50 1 3 50 1 4 50 2 3 50 2 5 10 3 4 50 3 5 25 4 5 20	1 3 4

Competição de chocolate

Nome do arquivo fonte: `chocolate.c`, `chocolate.cpp`, *ou* `chocolate.pas`

Carlos e Paula acabaram de ganhar um saco com bolinhas de chocolate. Como sabem que vão comer tudo muito rápido inventaram uma brincadeira:

- Eles vão comer de forma alternada, um depois o outro, sendo que sempre a Paula começa.
- A cada vez, só se pode comer de 1 a M bolinhas, sendo o M decidido pela mãe de Paula, de forma que não engasguem com o chocolate.
- Se um comeu K bolinhas em sua vez, o próximo não pode comer o mesmo tanto, tendo que comer um número de bolinhas distinto.
- Quem não puder mais jogar de maneira válida perde.

Um exemplo de partida para $M = 5$ e 20 bolinhas, onde Carlos ganhou:

Quem joga	Quantas comeu	Número de bolinhas restantes
-	-	20
Paula	5	15
Carlos	4	11
Paula	3	8
Carlos	4	4
Paula	2	2
Carlos	1	1

Observe que no final Carlos não poderia comer 2 bolinhas para ganhar, pois seria o mesmo que Paula comeu na vez anterior. Mas Paula também não pôde comer a última bolinha, pois Carlos havia comido apenas uma na rodada anterior, assim Paula ficou sem opção de jogada e perdeu.

Ambos são muito espertos e jogam de maneira ótima, de forma que se existe para um deles uma sequência de jogadas que garante a vitória independente da jogada do outro, essa pessoa jogará dessa forma.

Tarefa

Sua tarefa é determinar quem vai ganhar a brincadeira, se ambos jogam de forma ótima.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado).

A entrada consiste de uma linha contendo dois inteiros N ($2 \leq N \leq 10^6$) e M ($2 \leq M \leq 10^3$), sendo N o número de bolinhas de chocolate e M o número de bolinhas permitidas por vez.

Saída

Seu programa deve imprimir, na *saída padrão*, uma linha, contendo o nome do vencedor, como exemplificado abaixo.

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 50 pontos, $N \leq 50$ e $M \leq 5$.
- Em um conjunto de casos de teste que totaliza 80 pontos, $N \leq 10^4$ e $M \leq 100$.

Exemplos

Entrada	**Saída**
`5 3`	`Paula`

Entrada	**Saída**
`20 5`	`Carlos`

Entrada	**Saída**
`5 6`	`Paula`