Competidor(a): _____

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

OBI2018

Caderno de Tarefas

Modalidade **Programação** • **Nível Sênior** • Fase **Local**

17 de maio de 2018

A PROVA TEM DURAÇÃO DE **2** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

v1.0

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

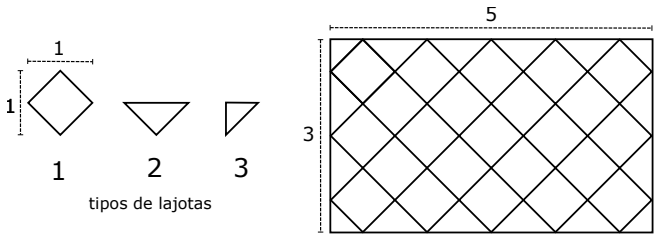
- Este caderno de tarefas é composto por 6 páginas (não contando a folha de rosto), numeradas de 1 a 6. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta 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.

Piso da escola

Nome do arquivo: "`piso.x`", *onde* `x` *deve ser* `cpp`, `pas`, `java`, `js`, `py2` *ou* `py3`

O colégio pretende trocar o piso de uma sala de aula e a diretora aproveitou a oportunidade para passar uma tarefa aos alunos. A sala tem o formato de um retângulo de largura L metros e comprimento C metros, onde L e C são números inteiros. A diretora precisa comprar lajotas de cerâmica para cobrir todo o piso da sala. Seria fácil calcular quantas lajotas seriam necessárias se cada lajota fosse um quadrado de 1 metro de lado. O problema é que a lajota que a diretora quer comprar é um quadrado que possui 1 metro de diagonal, não de lado. Além disso, ela quer preencher o piso da sala com as diagonais das lajotas alinhadas aos lados da sala, como na figura.

A loja vai fornecer lajotas do tipo 1: inteiras; do tipo 2, que correspondem à metade das do tipo 1, cortadas ao longo da diagonal; e lajotas do tipo 3, que correspondem à metade do tipo 2. Veja os três tipos de lajotas na figura.



Está muito claro que sempre serão necessárias 4 lajotas do tipo 3 para os cantos da sala. A tarefa que a diretora passou para os alunos é calcular o número de lajotas dos tipos 1 e 2 que serão necessárias. Na figura, para $L = 3$ e $C = 5$, foram necessárias 23 do tipo 1 e 12 do tipo 2.

Seu programa precisa computar, dados os valores de L e C, a quantidade de lajotas do tipo 1 e do tipo 2 necessárias.

Entrada

A primeira linha da entrada contém um inteiro L indicando a largura da sala. A segunda linha contém um inteiro C representando o comprimento da sala.

Saída

Imprima duas linhas na saída. A primeira deve conter um inteiro, representando o número de lajotas do tipo 1 necessárias. A segunda deve conter um inteiro, indicando o número de lajotas do tipo 2.

Restrições

- $1 \leq L, C \leq 100$

Exemplo de entrada 1	**Exemplo de saída 1**
3 5	23 12

Exemplo de entrada 2	**Exemplo de saída 2**
1 1	1 0

Figurinhas da Copa

Nome do arquivo: "`figurinhas.x`", *onde* `x` *deve ser* `cpp`, `pas`, `java`, `js`, `py2` *ou* `py3`

Em ano de Copa do Mundo de Futebol, o álbum de figurinhas oficial é sempre um grande sucesso entre crianças e também entre adultos. Para quem não conhece, o álbum contém espaços numerados de 1 a N para colar as figurinhas; cada figurinha, também numerada de 1 a N, é uma pequena foto de um jogador de uma das seleções que jogará a Copa do Mundo. O objetivo é colar todas as figurinhas nos respectivos espaços no álbum, de modo a *completar* o álbum (ou seja, não deixar nenhum espaço sem a correspondente figurinha).

Algumas figurinhas são *carimbadas* (efetivamente têm um carimbo impresso sobre a fotografia do jogador) e são mais raras, mais difíceis de conseguir.

As figurinhas são vendidas em envelopes fechados, de forma que o comprador não sabe quais figurinhas está comprando, e pode ocorrer de comprar uma figurinha que ele já tenha colado no álbum. Para ajudar os usuários, a empresa responsável pela venda do álbum e das figurinhas quer criar um aplicativo que permita gerenciar facilmente as figurinhas que faltam para completar o álbum.

Dados o número total de espaços e figurinhas do álbum (N), a lista das figurinhas carimbadas e uma lista das figurinhas já compradas (que pode conter figurinhas repetidas), sua tarefa é determinar quantas *figurinhas carimbadas* faltam para *completar* o álbum.

Entrada

A primeira linha contém três números inteiros N, C e M indicando respectivamente o número de figurinhas (e espaços) do álbum, o número de figurinhas carimbadas do álbum e o número de figurinhas já compradas. A segunda linha contém C números inteiros distintos X_i indicando as figurinhas carimbadas do álbum. A terceira linha contém M números inteiros Y_i indicando as figurinhas já compradas.

Saída

Seu programa deve produzir um inteiro representando o número de *figurinhas carimbadas* que falta para completar o álbum.

Restrições

- $1 \leq N \leq 100$
- $1 \leq C \leq N/2$
- $1 \leq M \leq 300$
- $1 \leq X_i, Y_i \leq N$

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

`10 2 5`
`4 7`
`7 1 2 8 3`
	`1`

Exemplo de entrada 2	**Exemplo de saída 2**

`10 2 6`
`4 7`
`7 1 8 4 9 3`
	`0`

Exemplo de entrada 3	**Exemplo de saída 3**
8 4 10 2 4 6 8 3 1 1 5 9 1 7 7 1 1	4

Ilhas

Nome do arquivo: "`ilhas.x`", *onde* `x` *deve ser* `cpp`, `pas`, `java`, `js`, `py2` *ou* `py3`

Os moradores das Ilhas Brasileiras Ocidentais (IBO) são assíduos jogadores do mais recente jogo *online*, *Magos e Guerreiros*. Tão competitivas se tornaram as partidas de Magos e Guerreiros na IBO, que a empresa criadora do jogo decidiu instalar em uma das ilhas um servidor dedicado apenas aos jogadores da IBO.

Entretanto, a empresa sabe que, se os jogadores acharem que o novo servidor é *injusto*, eles irão parar de jogar Magos e Guerreiros, gerando incontáveis perdas. Para avaliar se o novo servidor é justo, os jogadores vão comparar o desempenho do jogo na ilha que tem a conexão mais rápida e o desempenho na ilha que tem a conexão mais lenta com o novo servidor. Se a diferença de desempenho for muito grande, os residentes da ilha mais distante se sentirão injustiçados e abandonarão o jogo.

A conexão de internet da IBO funciona através de um sistema de cabos de fibra ótica. Pares de ilhas são conectados por cabos, e cada cabo toma um certo tempo (chamado de *ping*) para comunicar informação entre as duas partes. Quando duas ilhas se comunicam através de uma série de cabos (portanto, através de ilhas intermediárias), o *ping* entre elas é a soma dos *pings* de cada cabo no caminho. A rede da IBO foi implementada por ótimos programadores e, portanto, um par de ilhas sempre se comunica através do caminho com menor *ping* possível.

Dada a configuração da rede da IBO e a ilha em que a empresa deseja instalar o novo servidor, determine a diferença entre os *pings* da ilha com menor e maior *pings* até o servidor.

Entrada

A primeira linha contém N e M, o número de ilhas e o número de cabos de fibra ótica, respectivamente. As ilhas são numeradas de 1 a N. Cada uma das M linhas seguintes contém três inteiros U_i, V_i e P_i e descreve um cabo entre as ilhas U_i e V_i com *ping* P_i (note que cabos transmitem informação em ambas as direções). Finalmente, a última linha contém um inteiro S, o número da ilha em que o servidor será instalado.

Cada par de ilhas é conectado por no máximo um cabo de fibra ótica, e nenhum cabo conecta uma ilha a si mesma. É garantido que qualquer ilha consegue se comunicar com qualquer outra através de algum caminho de cabos de fibra ótica.

Saída

Seu programa deve produzir um inteiro representando a diferença entre o *ping* da ilha com maior *ping* até o servidor, e o da ilha com menor *ping* até o servidor. Note que a ilha em que o servidor se encontra não é considerada no cálculo do menor *ping*.

Restrições

- $2 \leq N \leq 1000$
- $N - 1 \leq M \leq 10^5$
- $1 \leq U_i \leq N$
- $1 \leq V_i \leq N$
- $1 \leq S \leq N$
- $1 \leq P_i \leq 1000$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 30 pontos, $2 \leq N \leq 100$ e $N - 1 \leq M \leq 1000$.

Exemplo de entrada 1	**Exemplo de saída 1**
4 5 2 1 5 1 3 4 2 3 6 4 2 8 3 4 12 1	9

Exemplo de entrada 2	**Exemplo de saída 2**
6 11 1 2 3 6 1 9 2 6 10 2 3 8 5 3 3 4 3 2 2 4 12 6 4 1 4 5 9 1 5 16 5 6 5 5	11