Competidor(a): _____

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

OBI2019

Caderno de Tarefas

Modalidade **Programação** • **Nível Sênior** • Fase **Nacional**

21 de setembro de 2019

A PROVA TEM DURAÇÃO DE **5** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

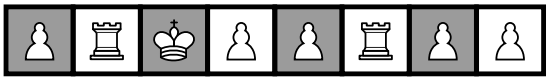
- Este caderno de tarefas é composto por 8 páginas (não contando a folha de rosto), numeradas de 1 a 8. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta 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 tarefas 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.py*; soluções na linguagem Python 3 devem ser arquivos com sufixo *_py3.py*; 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.

Xadrez Aleatório

Nome do arquivo: `xadrez.c`, `xadrez.cpp`, `xadrez.pas`, `xadrez.java`, `xadrez.js`, `xadrez_py2.py` *ou* `xadrez_py3.py`

Xadrez Aleatório de Fischer, ou Xadrez 960, é uma variante do jogo de Xadrez que usa exatamente as mesmas regras com uma única exceção, a posição inicial das peças é sorteada antes do jogo. As peças da primeira linha do tabuleiro podem estar em qualquer posição desde que respeitem duas restrições: o rei deve estar entre as duas torres; e os dois bispos devem estar em casas de cores opostas. Como você já deve ter desconfiado, o número de posições iniciais válidas nessa variante do Xadrez é 960.

Neste problema queremos contar o número de posições iniciais válidas numa outra variante, bem mais simples. A dimensão do tabuleiro não é mais fixa. Para qualquer dimensão, a primeira linha do tabuleiro vai conter apenas três tipos de peças: rei, torre e peão. Haverá sempre exatamente um rei e no máximo duas torres. O número de peões será a dimensão menos a soma do número das demais peças. Se o número de torres for dois, então o rei deve estar entre as duas torres. A figura abaixo mostra uma posição inicial válida para $N = 8$.



Entrada

A entrada consiste de apenas uma linha contendo dois inteiros, N e T, representando, respectivamente, a dimensão do tabuleiro e o número de torres.

Saída

Seu programa deve imprimir uma linha contendo um inteiro indicando o número de posições iniciais válidas.

Restrições

- $2 \leq N \leq 1000$
- $0 \leq T \leq 2$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 10 pontos, $T = 0$.
- Para um conjunto de casos de testes valendo outros 20 pontos, $T = 1$.
- Para um conjunto de casos de testes valendo outros 20 pontos, $N = 4$.

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

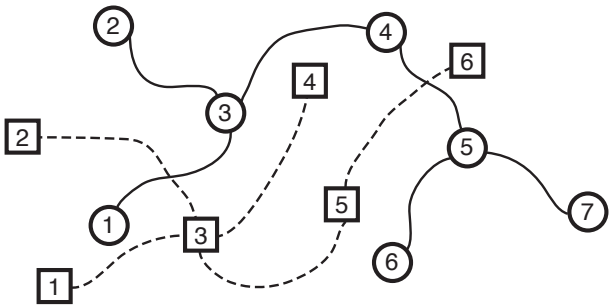
Exemplo de entrada 2	**Exemplo de saída 2**
`8 1`	`56`

Exemplo de entrada 3	**Exemplo de saída 3**
`213 2`	`1587986`

Metrô da Nlogônia

Nome do arquivo: `metro.c`, `metro.cpp`, `metro.pas`, `metro.java`, `metro.js`, `metro_py2.py` *ou* `metro_py3.py`

Há dois sistemas de metrô na capital da Nlogônia, operados por duas empresas diferentes. O dois sistemas, denominados Círculo e Quadrado, são independentes e não conectados entre si, ou seja, não há nenhuma estação em comum e nenhum trilho em comum. Em cada sistema há exatamente um caminho possível entre duas estações quaisquer, possivelmente passando por outras estações do sistema. A figura abaixo mostra uma representação de dois sistemas de metrô independentes, similares ao metrô da capital da Nlogônia. Apropriadamente, no sistema Círculo as estações são representadas por círculos, e no sistema Quadrado as estações são representadas por quadrados.



Vamos chamar de *diâmetro do sistema* de metrô o maior número de estações no trajeto entre qualquer par de estações do sistema. Assim, o diâmetro do sistema Círculo na figura acima é cinco (trajeto `2-3-4-5-7` por exemplo) e o diâmetro do sistema Quadrado é quatro (trajeto `4-3-5-6` por exemplo).

O rei da Nlogônia decidiu que os dois sistemas existentes devem ser integrados, para facilitar a vida dos usuários. A integração vai ser implementada através da construção de um único novo trecho de metrô ligando exatamente um par de estações existentes (uma estação do sistema Círculo e uma estação do sistema Quadrado). O rei determinou ainda que o diâmetro do sistema integrado seja o menor possível.

Você pode ajudar a planejar a integração dos sistemas? Dadas as descrições dos dois sistemas, sua tarefa é determinar qual par de estações deve ser ligado para realizar a integração como desejada pelo rei.

Entrada

A primeira linha contém dois inteiros N e M, indicando respectivamente o número de estações do sistema Círculo e do sistema Quadrado. No sistema Círculo as estações são identificadas por números de 1 a N e no sistema Quadrado as estações são identificadas por números de 1 a M. Cada uma das $N-1$ linhas seguintes descreve as ligações entre estações do sistema Círculo e contém dois inteiros A e B indicando que existe uma ligação entre as estações A e B. Cada uma das $M-1$ linhas seguintes descreve as ligações entre estações do sistema Quadrado e contém dois inteiros X e Y indicando que existe uma ligação entre as estações X e Y.

Saída

Seu programa deve produzir dois inteiros, o primeiro representando uma estação do sistema Círculo e o segundo representando uma estação do sistema Quadrado. Se houver mais de um par possível, indique qualquer um entre os possíveis.

Restrições

- $2 \leq N, M \leq 10^5$
- $1 \leq A, B \leq N$

- $1 \leq X, Y \leq M$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 10 pontos, $N, M \leq 100$.
- Para um conjunto de casos de testes valendo outros 20 pontos, $N, M \leq 1000$.

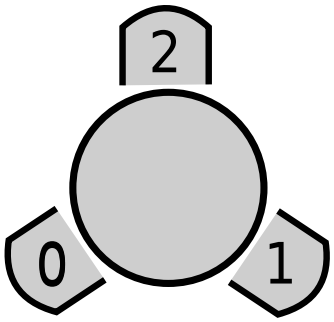
Exemplo de entrada 1	**Exemplo de saída 1**
7 6 1 3 3 2 3 4 4 5 5 7 6 5 1 3 2 3 3 4 3 5 5 6	4 3

Exemplo de entrada 2	**Exemplo de saída 2**
3 4 1 2 2 3 1 2 2 3 3 4	2 2

Mesa redonda

Nome do arquivo: `mesa.c`, `mesa.cpp`, `mesa.pas`, `mesa.java`, `mesa.js`, `mesa_py2.py` *ou* `mesa_py3.py`

Ana, Beatriz e Carolina sempre saem juntas para tomar café numa padaria onde as mesas são circulares e têm três cadeiras numeradas 0, 1 e 2, no sentido anti-horário, como ilustrado na figura ao lado.



Elas gostam de decidir quem vai sentar em qual cadeira com uma brincadeira gerando números aleatórios nos seus celulares. Primeiro Ana sorteia um número inteiro A e, começando da cadeira 1, seguindo no sentido anti-horário, conta A cadeiras e senta na cadeira em que a contagem terminar. Depois Beatriz sorteia um número B e faz a mesma coisa: começando da cadeira 1, no sentido anti-horário, conta B cadeiras. Se a cadeira final estiver livre, Beatriz senta nela. Caso seja a cadeira onde Ana está sentada, então Beatriz senta na próxima cadeira no sentido anti-horário. Claro, ao final, Carolina senta na cadeira que estiver livre.

Por exemplo, se Ana sortear 8, ela vai contar $[1, 2, 0, 1, 2, 0, 1, 2]$ e sentar na cadeira 2. Depois, se Beatriz sortear 6, ela vai contar $[1, 2, 0, 1, 2, 0]$ e sentar na cadeira 0. Assim, Carolina senta na cadeira 1. Num outro exemplo, se Ana sortear 3, ela vai contar $[1, 2, 0]$ e sentar na cadeira 0. Depois, se Beatriz sortear 9, ela vai contar $[1, 2, 0, 1, 2, 0, 1, 2, 0]$ e, como Ana já está sentada na cadeira 0, Beatriz senta na cadeira 1. Dessa forma, Carolina senta na cadeira 2.

Neste problema, dados os números sorteados por Ana e Beatriz, seu programa deve imprimir o número da cadeira onde Carolina vai sentar.

Entrada

A primeira linha da entrada contém um inteiro A representando o número sorteado por Ana. A segunda linha da entrada contém um inteiro B representando o número sorteado por Beatriz.

Saída

Seu programa deve imprimir uma linha contendo um número inteiro indicando a cadeira onde Carolina vai sentar.

Restrições

- $1 \leq A \leq 1000$
- $1 \leq B \leq 1000$

Exemplo de entrada 1	**Exemplo de saída 1**
8 6	1

Exemplo de entrada 2	**Exemplo de saída 2**
3 9	2

Computador

Nome do arquivo: `computador.c`, `computador.cpp`, `computador.pas`, `computador.java`, `computador.js`, `computador_py2.py` *ou* `computador_py3.py`

Uma grande empresa está construindo uma nova arquitetura de computadores que permita a execução eficiente de duas instruções especiais de soma. O computador possui N posições de memória, endereçadas de 1 a N, e cada posição pode guardar um inteiro maior ou igual a zero. Inicialmente, todas as posições contêm o valor zero. As instruções especiais de soma são:

- FRENTE i V: Dado o endereço i, $1 \leq i \leq N$, e um valor positivo V, o computador deve somar V na posição i, $V-1$ em $i+1$, $V-2$ em $i+2$, etc, enquanto o valor a ser somando for maior do que zero e a posição for menor ou igual a N;
- TRÁS i V: Dado o endereço i, $1 \leq i \leq N$, e um valor positivo V, o computador deve somar V na posição i, $V-1$ em $i-1$, $V-2$ em $i-2$, etc, enquanto o valor a ser somando for maior do que zero e a posição for maior ou igual a 1.

Por exemplo, para $N = 16$, uma possível sequência de instruções é dada abaixo:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FRENTE 4 8

0	0	0	8	7	6	5	4	3	2	1	0	0	0	0	0

TRÁS 16 3

0	0	0	8	7	6	5	4	3	2	1	0	0	1	2	3

TRÁS 2 12

11	12	0	8	7	6	5	4	3	2	1	0	0	1	2	3

FRENTE 8 7

11	12	0	8	7	6	5	11	9	7	5	3	2	2	2	3

Além disso, o computador possui a instrução IMPRIME i, que deve imprimir na saída o valor atual armazenado na posição i da memória.

Dados N e uma sequência de M instruções, seu programa deve imprimir, para cada instrução do tipo IMPRIME i, uma linha contendo o valor armazenado na posição de memória i no instante da execução da instrução.

Entrada

A primeira linha da entrada contém dois inteiros N e M, representando o número de posições de memória e o número de instruções, respectivamente. As M linhas seguintes contêm, cada uma, a descrição de uma instrução em uma de três formas possíveis: 1 I V, representando FRENTE I V; 2 I V, representando TRÁS I V; e 3 I, representando IMPRIME I.

Saída

Para cada instrução do tipo IMPRIME i, seu programa deve imprimir uma linha contendo um inteiro representando o valor armazenado na posição de memória i no instante da execução da instrução.

Restrições

- $1 \leq N \leq 200000$;
- $1 \leq M \leq 200000$;
- $1 \leq I \leq N$;
- $1 \leq V \leq 200000$;
- Ao menos uma instrução será do tipo 3.

Informações sobre a pontuação

- Em um conjunto de casos de teste somando 20 pontos, $N \leq 10000$, $M \leq 10000$ e $V \leq 10000$;

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
16 7 1 4 8 2 16 3 3 14 2 2 12 1 8 7 3 10 3 14	1 7 2

Exemplo de entrada 2	**Exemplo de saída 2**
200000 2 1 2345 193290 3 112230	83405

Etiquetas

Nome do arquivo: `etiquetas.c`, `etiquetas.cpp`, `etiquetas.pas`, `etiquetas.java`, `etiquetas.js`, `etiquetas_py2.py` *ou* `etiquetas_py3.py`

Uma fita esticada horizontalmente é composta de N quadrados de dimensão 1×1, cada um deles contendo um número inteiro anotado. Temos também K etiquetas retangulares idênticas, de dimensão $1 \times C$, onde C é um inteiro. Nosso objetivo é colar todas as etiquetas sobre a fita de modo que a soma dos inteiros que não estiverem cobertos por nenhuma etiqueta ao final seja a máxima possível. Cada etiqueta deve ser colada na horizontal, ao longo da fita. Duas etiquetas não podem estar sobrepostas e cada quadrado da fita deve estar ou totalmente coberto por uma etiqueta, ou totalmente descoberto.

Entrada

A primeira linha da entrada contém três inteiros N, K e C, representando, respectivamente, o comprimento da fita, o número de etiquetas e o comprimento das etiquetas. A segunda linha da entrada contém N inteiros A indicando a sequência de números anotados nos quadrados da fita.

Saída

Seu programa deve imprimir uma linha contendo um inteiro indicando a soma máxima possível de inteiros descobertos na fita depois que todas as etiquetas sejam coladas seguindo as condições do enunciado.

Restrições

- $1 \leq N \leq 10000$
- $-10000 \leq A \leq 10000$
- $1 \leq K \leq 10000$
- $1 \leq C \leq 10000$
- $K \times C \leq N$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 10 pontos, os números no vetor estão ordenados crescentemente.
- Para um conjunto de casos de testes valendo outros 10 pontos, $C = 1$.
- Para um conjunto de casos de testes valendo outros 20 pontos, $K = 1$.
- Para um conjunto de casos de testes valendo outros 20 pontos, $K, N \leq 100$.

Exemplo de entrada 1	**Exemplo de saída 1**
`12 2 3` `1 22 4 -8 9 2 10 -1 5 5 32 -11`	`58`

Exemplo de entrada 2	**Exemplo de saída 2**
`1 1 1` `10000`	`0`