Competidor(a): _____

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

OBI2020

Caderno de Tarefas

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

15 e 16 de junho de 2020

A PROVA TEM DURAÇÃO DE **2** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

v1.0

Instruções

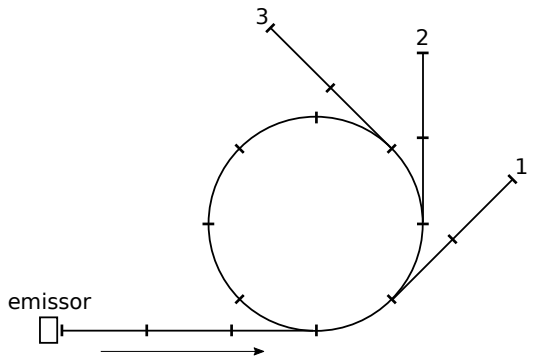
LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 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 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.

Acelerador de partículas

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

A universidade está inaugurando um grande acelerador de partículas, com um emissor e três sensores, numerados 1, 2 e 3. Uma partícula, após sair do emissor, entra no acelerador onde pode dar várias voltas sendo acelerada a velocidades muito altas. Num determinado momento, a partícula sai do acelerador por uma das três saídas, atingindo um dos sensores. A figura mostra o caminho por onde as partículas trafegam, com uma graduação de 1 quilômetro. Por exemplo, do emissor até o acelerador são 3 quilômetros e a circunferência do acelerador tem 8 quilômetros.



Neste problema, será dada a distância total, em quilômetros, percorrida por uma certa partícula trafegando do emissor até algum sensor e seu programa deve determinar qual sensor foi atingido pela partícula. Por exemplo, veja que se a distância total for 23 quilômetros, então a partícula tem que ter atingido o sensor 2.

Entrada

A entrada consiste de apenas uma linha contendo um inteiro D, representando a distância total percorrida pela partícula.

Saída

Seu programa deve imprimir uma linha contendo um inteiro, representando o número do sensor que a partícula atingiu.

Restrições

- $6 \leq D \leq 800008$. D sempre será a distância total percorrida entre o emissor e algum sensor.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
23	2

Exemplo de entrada 2	**Exemplo de saída 2**
6	1

Exemplo de entrada 3	**Exemplo de saída 3**
9192	3

Fissura Perigosa

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

A erupção do vulcão Kilauea em 2018 no Havaí atraiu a atenção de todo o mundo. Inicialmente a força da erupção era menor e a lava avançou para o sul com relativamente poucos danos. Após algumas semanas, porém, a fissura 8 começou a jorrar com mais força e a lava avançou também para o norte trazendo muita destruição.

Você está ajudando na implementação de um sistema para simular a área por onde a lava avançaria, em função da força da erupção. O mapa será representado simplificadamente por uma matriz quadrada de caracteres, de `1` a `9`, indicando a altitude do terreno em cada posição da matriz. Vamos considerar que a fissura 8, por onde a erupção se inicia, está sempre na posição do canto superior esquerdo da matriz. Dada a força da erupção, que será um valor inteiro, de 0 a 9, seu programa deve imprimir a matriz de caracteres representando o avanço final da lava. Se a lava consegue invadir uma posição da matriz, o caractere naquela posição deve ser trocado por um asterisco ('`*`'). Uma posição será invadida pela lava se seu valor for menor ou igual à força da erupção e

- for a posição inicial; ou
- estiver adjacente, ortogonalmente (abaixo, acima, à esquerda ou à direita), a uma posição invadida.

A figura abaixo mostra um exemplo de mapa e o avanço final da lava para quatro forças de erupção: 1, 3, 6 e 8, respectivamente da esquerda para a direita.

```
27755478    *7755478    *7755478    ********
29985439    *9985439    *9985439    *99****9
34899989    *4899989    **899989    ***999*9
22115569    ****5569    *******9    *******9
66736689    667*6689    **7***89    *******9
99886555    99886555    9988****    99******
44433399    44433399    ******99    ******99
99986991    99986991    9998*991    999**991
```

Entrada

A primeira linha da entrada contém dois inteiros N e F representando, respectivamente o número de linhas (que é igual ao de colunas) da matriz e a força da erupção. Cada uma das N linhas seguintes contém uma string de N caracteres, entre `1` e `9`, indicando o mapa de entrada.

Saída

Seu programa deve imprimir N linhas contendo, cada uma, N caracteres representando o avanço final da lava de acordo com o enunciado.

Restrições

- $1 \leq N \leq 500$
- $0 \leq F \leq 9$

Informações sobre a pontuação

- Em um conjunto de casos de teste somando 20 pontos, $N \leq 10$.
- Em um conjunto de casos de teste somando 20 pontos, $10 < N \leq 100$.
- Em um conjunto de casos de teste somando 60 pontos, nenhuma restrição adicional.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
8 6 27755478 29985439 34899989 22115569 66736689 99886555 44433399 99986991	*7755478 *9985439 **899989 *******9 **7***89 9988**** ******99 9998*991

Exemplo de entrada 2	**Exemplo de saída 2**
5 4 25679 35234 17182 39993 11223	*5679 *5*** *7*8* *999* *****

Exemplo de entrada 3	**Exemplo de saída 3**
2 8 91 11	91 11

Bingo!

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

O grande prêmio do Bingo de São João será um carro zero-quilômetro. Todo mundo quer ser o primeiro a completar sua cartela, claro. São N cartelas identificadas de 1 até N que contêm, cada uma, K números distintos entre os números naturais de 1 até U, para $K < U$. Um número, claro, pode aparecer em mais de uma cartela e duas cartelas podem até ser iguais, ter o mesmo conjunto de números. Justamente por isso, veja que pode acontecer empate com mais de uma cartela sendo completada no mesmo instante.

Neste problema, serão dados na entrada os conjuntos de números de todas as cartelas e a sequência de números sorteados, que será uma permutação dos naturais de 1 até U. Seu programa deve determinar qual ou quais cartelas vão ser completadas primeiro e ganhar o carro.

Por exemplo, para $N = 4$, $K = 5$ e $U = 10$, com as cartelas dadas pela tabela abaixo, se a sequência de números sorteados for $[7, 3, 5, 2, 6, 1, 9, 10, 4, 8]$, então haverá uma cartela vencedora, a número 3.

número da cartela					
1	3	10	8	7	2
2	4	1	7	10	9
3	9	1	5	3	6
4	6	8	1	5	7

Entrada

A primeira linha da entrada contém três inteiros N, K e U representando respectivamente o número de cartelas, quantos números cada cartela contém e o maior natural que pode ocorrer numa cartela. As N linhas seguintes contêm, cada uma, K inteiros distintos C_i, para $1 \leq i \leq K$, representando o conjunto de números de cada cartela, da cartela 1 até a N. A última linha da entrada contém U inteiros indicando a sequência de números sorteados, uma permutação dos naturais entre 1 e U.

Saída

Seu programa deve imprimir uma linha contendo os números identificadores das cartelas vencedoras do carro, em ordem crescente.

Restrições

- $1 \leq N \leq 1000$
- $1 \leq K \leq 1000$
- $1 \leq U \leq 10000$

Informações sobre a pontuação

- Em um conjunto de casos de teste somando 15 pontos, $N \leq 100, K \leq 50, U \leq 100$, e apenas uma cartela é vencedora.
- Em um conjunto de casos de teste somando 15 pontos, $N \leq 100, K \leq 50, U \leq 100$.
- Em um conjunto de casos de teste somando 30 pontos, $N \leq 100, K \leq 500, U \leq 1000$.
- Em um conjunto de casos de teste somando 40 pontos, nenhuma restrição adicional.

Exemplos

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

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

Paciente Zero

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

Quando ocorre uma epidemia por uma nova espécie de vírus, uma das tarefas dos infectologistas é determinar o *Paciente Zero*, ou seja, a pessoa que foi infectada primeiro pelo novo vírus. A primeira infecção é geralmente causada por transmissão do novo vírus de algum hospedeiro não humano como morcego, rato, ou macaco, para um humano. Esse primeiro humano infectado então infecta outras pessoas, que por sua vez infectam ainda outras pessoas, e assim surge a epidemia.

A procura pelo Paciente Zero é feita através de um minucioso trabalho de entrevistas com infectados para determinar quem contaminou quem desde o início da epidemia. Vamos chamar de "cadeia de infecção" a sequência de infecção causada por uma pessoa. Por exemplo, se as entrevistas determinarem que João infectou Clarisse, Clarisse infectou Silvia, e Silvia infectou Rafael, a cadeia de infecção de João é Clarisse $\rightarrow$ Silvia $\rightarrow$ Rafael. O Paciente Zero é definido como a pessoa infectada com o vírus que não foi infectada por nenhum outro humano, ou seja, não faz parte da cadeia de infeção de nenhuma outra pessoa. Às vezes não é possível determinar um único Paciente Zero, e nesse caso um grupo de pessoas são designadas como Pacientes Zero.

Nesta tarefa você deve determinar o Paciente ou Pacientes Zero a partir das cadeias de infecção determinadas pelos infectologistas.

Entrada

A primeira linha da entrada contém dois números inteiros N e C, respectivamente o total de pessoas infectadas e o número de cadeias de transmissão. As pessoas são identificados por números inteiros de 1 a N. Cada uma das C linhas seguintes contém a informação sobre uma cadeia de transmissão. A linha inicia com um número P, o identificador da pessoa infectante, seguido de um número I, o total pessoas nessa cadeia de transmissão, seguido de I inteiros X_i identificando as pessoas na cadeia de transmissão.

Cada pessoa consta, no máximo, de uma cadeia de transmissão. Em outras palavras, se um pessoa aparece uma cadeia de transmissão, ela não aparece em nenhuma outra cadeia de transmissão.

Saída

Se houver apenas um Paciente Zero, seu programa deve produzir na saída uma linha contendo o número identificador do Paciente Zero. Se houver K Pacientes Zero, seu programa deve produzir na saída K linhas, cada uma contendo o identificador um Paciente Zero distinto, em ordem crescente do número de identificação dos pacientes.

Restrições

- $2 \leq N \leq 1000$
- $1 \leq C \leq N - 1$
- $1 \leq P \leq N$
- $1 \leq I \leq N - 1$
- $1 \leq X_i \leq N$ para $1 \leq i \leq I$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 20 pontos, $2 \leq N \leq 10$ e há apenas um Paciente Zero.
- Para um conjunto de casos de testes valendo 40 pontos, $10 < N \leq 500$.
- Para um conjunto de casos de testes valendo 40 pontos, nenhuma restrição adicional.

Exemplos

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

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