Competidor(a): _____

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

OBI2020

Caderno de Tarefas

Modalidade **Programação** • **Nível 2** • 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

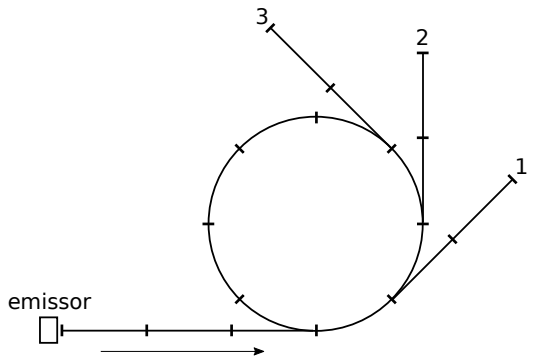
LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- 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

Promoção de Primeira

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

O reino da Linearlândia possui N cidades espalhadas por seu vasto território, sendo que $N - 1$ pares distintos de cidades estão ligados diretamente por uma rodovia bi-direcional. Esses pares foram escolhidos de forma que existe exatamente um caminho entre qualquer par de cidades, possivelmente passando por outras cidades no meio do caminho. Cada rodovia da Linearlândia é servida por uma linha de ônibus, que faz viagens de ida e volta entre as duas cidades, operada por apenas uma empresa, como manda a lei determinada pelo Rei. O problema é que existem apenas duas empresas de ônibus: a RoyalBus e a ImperialBus.

Cada viagem entre duas cidades ligadas diretamente por uma rodovia custa uma passagem da empresa que opera aquela linha. Ao chegar numa cidade, se o passageiro quiser prosseguir viagem para outra cidade, ele tem que desembarcar, entrar em outro ônibus e pagar outra passagem. Só que o Rei determinou, para o feriadão anual de celebração da Linearidade Real, uma estranha promoção: sempre que o passageiro entrar no ônibus de uma empresa ele não precisa pagar a passagem se sua viagem imediatamente anterior foi pela outra empresa. Quer dizer, se o caminho alterna entre a RoyalBus e a ImperialBus, só é preciso pagar uma passagem, a primeira.

Neste problema, dada a descrição da malha de rodovias da Linearlândia, seu programa deve computar o número máximo de cidades num caminho, começando em qualquer cidade, para o qual é preciso pagar apenas uma passagem para ir da cidade inicial até a cidade final do caminho. O número de cidades no caminho inclui a cidade inicial e a cidade final.

Entrada

A primeira linha da entrada contém um inteiro N, representando o número de cidades da Linearlândia. As cidades são numeradas de 1 até N. As $N - 1$ linhas seguintes contêm, cada uma, três inteiros A, B e E, indicando que existe uma rodovia entre as cidades A e B e que a linha de ônibus entre elas é operado pela empresa E (0 para RoyalBus, 1 para ImperialBus).

Saída

Seu programa deve imprimir uma linha contendo um inteiro representando o número máximo de cidades num caminho para o qual é preciso pagar apenas uma passagem durante a celebração da Linearidade Real.

Restrições

- $2 \leq N \leq 50000$
- $1 \leq A \leq N$
- $1 \leq B \leq N$
- $0 \leq E \leq 1$

Informações sobre a pontuação

- Em um conjunto de casos de teste somando 20 pontos, o número máximo de rodovias chegando em qualquer cidade é dois. Quer dizer, a malha é um longo caminho passando por todas as cidades.
- Em um conjunto de casos de teste somando 20 pontos, $N \leq 10^3$.
- Em um conjunto de casos de teste somando 20 pontos, $10^3 < N \leq 10^4$.
- Em um conjunto de casos de teste somando 40 pontos, nenhuma restrição adicional.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
8 3 1 0 2 7 0 6 8 1 1 4 1 5 4 1 4 7 1 7 6 0	4

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

Exemplo de entrada 3	**Exemplo de saída 3**
18 13 16 0 16 15 1 16 12 0 14 12 0 12 8 1 1 18 0 1 3 0 2 3 1 3 8 1 11 17 1 17 10 1 8 17 0 6 7 0 9 7 0 5 7 1 4 5 0 8 5 1	6

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

Pandemia

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

Um grupo de amigos, preocupados por ter que prestar o ENEM este ano, resolveu iniciar o ano fazendo reuniões de estudo. Mas eles não esperavam que uma epidemia com um novo vírus ocorresse na região em que moravam. Nessa epidemia específica, os sintomas da doença aparecem muitos dias depois do contágio, mas mesmo sem sintomas uma pessoa infectada infecta todos com quem tenha o mínimo contato.

O grupo de amigos também não sabia que um deles havia sido infectado, sem saber, por pessoas de fora do grupo, o que fez a infecção se espalhar pelos amigos do grupo. Felizmente todos os amigos infectados se recuperaram e passam bem.

Muitas reuniões de estudo aconteceram, mas nem todos os amigos participaram de todas as reuniões.

Você receberá a informação de quais amigos participaram de cada reunião. Além disso, você receberá também a informação de qual amigo participou de reunião do grupo após ter sido infectado por pessoas de fora do grupo, e em qual reunião isso ocorreu. Você deve assumir que:

1. todos os amigos que participaram de reunião em que ao menos um deles estava infectado também foram infectados.
2. o único amigo infectado por pessoas de fora do grupo é o que foi informado. No caso de todos os outros amigos que foram infectados a infecção aconteceu em reunião do grupo.

Escreva um programa para determinar quantos amigos, ao final da sequência de reuniões, foram infectados.

Entrada

A primeira linha da entrada contém dois números inteiros N, M, respectivamente o total de amigos do grupo e o total de dias em que houve reunião. Os amigos são identificados por números inteiros de 1 a N, as reuniões são identificadas por números inteiros de 1 (primeira reunião) a M (última reunião). A segunda linha contém dois números inteiros I e R, respectivamente o identificador do amigo que foi infectado por pessoas de fora do grupo e o número da primeira reunião em que ele participou infectado. Cada uma das M linhas seguintes contém a informação dos participantes de uma reunião, em sequência; ou seja, a primeira linha descreve os participantes da reunião 1, a segunda linha descreve os participantes da reunião 2 e assim por diante. Cada uma dessas linhas inicia com um número A, o total de amigos que participaram dessa reunião, seguido de A inteiros P_i identificando cada amigo participante da reunião.

Saída

Seu programa deve produzir um inteiro representando o número total de amigos infectados ao final do mês.

Restrições

- $2 \leq N \leq 1000$
- $2 \leq M \leq 1000$
- $1 \leq I \leq N$
- $1 \leq R \leq M$
- $1 \leq A \leq N$
- $1 \leq P_i \leq N$ para $1 \leq i \leq A$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 20 pontos, $N \leq 10$ e $M \leq 10$.
- Para um conjunto de casos de testes valendo 60 pontos, $10 < N \leq 500$ e $10 < M \leq 500$.
- Para um conjunto de casos de testes valendo 20 pontos, nenhuma restrição adicional.

Exemplos

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

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

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