OBI2017

Caderno de Tarefas

Modalidade **Universitária** • Fase **2**

9 de junho de 2017

A PROVA TEM DURAÇÃO DE **2** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

v1.0

Instruções

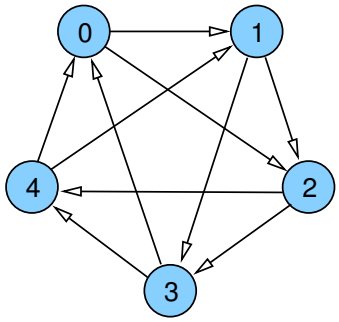
LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 7 páginas (não contando a folha de rosto), numeradas de 1 a 7. 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.

Dario e Xerxes

Nome do arquivo: `xerxes.c`, `xerxes.cpp`, `xerxes.pas`, `xerxes.java`, `xerxes.js`, `xerxes.py2` *ou* `xerxes.py3`

A brincadeira da Pedra, Papel e Tesoura, muita gente conhece. Mas dá para fazer uma mais legal com cinco opções e não só três! Dois jogadores, `dario` e `xerxes`, jogam uma partida com N rodadas. Em cada rodada os jogadores escolhem uma "mão" entre cinco opções, que vamos representar aqui com os números 0, 1, 2, 3 e 4. A figura define exatamente quem ganha a rodada. Por exemplo, se `dario` escolheu 0 e `xerxes` escolheu 3, então `xerxes` ganha a rodada, pois existe uma seta na figura indo de 3 para 0.



Depois de N rodadas, o vencedor da partida é o jogador que ganhou mais rodadas. O número N será sempre ímpar, para não haver empate na partida. Vamos também considerar que os jogadores nunca escolhem a mesma mão numa rodada, para não haver empate na rodada. Você deve escrever um programa que determine quem venceu a partida, se foi `dario` ou `xerxes`.

Entrada

A primeira linha da entrada contém um inteiro N, o número de rodadas na partida. Cada uma das N linhas seguintes contém dois inteiros D e X, representando a mão que os jogadores `dario` e `xerxes`, respectivamente, jogaram em uma rodada.

Saída

Seu programa deve imprimir uma linha contendo o nome do jogador que venceu a partida: `dario` ou `xerxes`. Todas as letras devem ser minúsculas, sem nenhum acento!

Restrições

- $1 \leq N \leq 999$, N é impar
- $0 \leq D, X \leq 4$ e $D \neq X$

Exemplos

Entrada	**Saída**
3 1 3 4 2 0 2	dario

Entrada	**Saída**
1 3 1	xerxes

Mapa

Nome do arquivo: `mapa.c`, `mapa.cpp`, `mapa.pas`, `mapa.java`, `mapa.js`, `mapa.py2` *ou* `mapa.py3`

Harry ganhou um mapa mágico no qual ele pode visualizar o trajeto realizado por seus amigos. Ele agora precisa de sua colaboração para, com a ajuda do mapa, determinar onde Hermione se encontra.

O mapa tem L linhas e C colunas de caracteres, que podem ser '.' (ponto), a letra 'o' (minúscula) ou a letra 'H' (maiúscula). A posição inicial de Hermione no mapa é indicada pela letra 'o', que aparece exatamente uma vez no mapa. A letra 'H' indica uma posição em que Hermione pode ter passado – o mapa é impreciso, e nem toda letra 'H' no mapa representa realmente uma posição pela qual Hermione passou. Mas todas as posições pelas quais Hermione passou são representadas pela letra 'H' no mapa.

A partir da posição inicial de Hermione, Harry sabe determinar a posição atual de sua amiga, apesar da imprecisão do mapa, porque eles combinaram que Hermione somente se moveria de forma que seu movimento apareceria no mapa como estritamente horizontal ou estritamente vertical (nunca diagonal). Além disso, Hermione combinou que não se moveria de forma a deixar que Harry tivesse dúvidas sobre seu caminho (por exemplo, Hermione não passa duas vezes pela mesma posição). Considere o mapa abaixo, com 6 linhas e 7 colunas:

	1	2	3	4	5	6	7
1	.	.	.	**H**	**H**	**H**	.
2	H	H	H	.	.	.	**H**
3	H	.	H	H	H	.	.
4	H	.	.	.	H	H	.
5	H	.	o	.	.	.	.
6	H	H	H	.	.	**H**	**H**

A posição inicial de Hermione no mapa é (5,3), e sua posição atual é (4,6). As posições marcadas em negrito ('**H**') são erros no mapa.

Dado um mapa e a posição inicial de Herminone, você deve escrever um programa para determinar a posição atual de Herminone.

Entrada

A primeira linha contém dois números inteiros L e C, indicando respectivamente o número de linhas e o número de colunas. Cada uma das seguintes L linhas contém C caracteres.

Saída

Seu programa deve produzir uma única linha na saída, contendo dois números inteiros: o número da linha e o número da coluna da posição atual de Hermione.

Restrições

- $2 \leq L \leq 100$
- $2 \leq C \leq 100$
- Apenas os caracteres '.', 'o' e 'H' aparecem no mapa.
- A letra 'o' aparece exatamente uma vez no mapa.
- A letra 'H' aparece ao menos uma vez no mapa.
- O caminho de Hermione está totalmente contido no mapa.
- Na posição da letra 'o' no mapa, há apenas uma letra 'H' como vizinho imediato na vertical ou horizontal.

- Na posição atual de Hermione no mapa, há apenas uma letra 'H' como vizinho imediato na vertical ou horizontal.
- Em cada uma das posições intermediárias do caminho de Hermione, há exatamente duas letras 'H' como vizinhas imediatas na vertical ou horizontal.

Informações sobre a pontuação

- Em um conjunto de casos de teste somando 20 pontos, $N \leq 8$

Exemplos

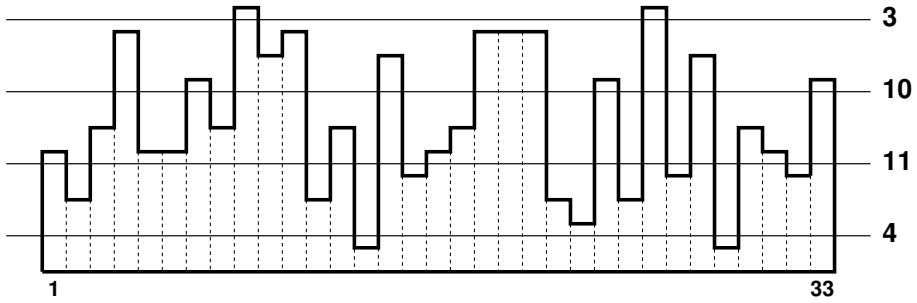
Entrada	**Saída**
3 4 HHHH H... o.HH	1 4

Entrada	**Saída**
6 7 ...HHH. HHH.... H.HHH.. H...HH. H.o.... HHH.HH.	4 6

Cortando o Papel

Nome do arquivo: `papel.c`, `papel.cpp`, `papel.pas`, `papel.java`, `papel.js`, `papel.py2` *ou* `papel.py3`

Uma folha de papel é composta de uma sequência de retângulos com diferentes alturas mas com larguras fixas, tal que as bases dos retângulos estão assentadas em uma linha horizontal. A figura ilustra uma folha exemplo com 33 retângulos. Nós gostaríamos de fazer um único corte horizontal, com a ajuda de um estilete e uma régua, que maximize o número resultante de pedaços separados pelo corte. A figura mostra quatro diferentes cortes que resultariam, respectivamente, em $4, 11, 10$ e 3 pedaços.



Entrada

A primeira linha da entrada contém um inteiro N, representando o número de retângulos na folha de papel. A segunda linha contém N inteiros A_i, $1 \leq i \leq N$, representando a sequência de alturas dos retângulos.

Saída

Seu programa deve imprimir uma linha contendo um inteiro representando o número máximo de pedaços possível, com um único corte horizontal.

Restrições

- $1 \leq N \leq 10^5$
- $1 \leq A_i \leq 10^9$, para $1 \leq i \leq N$

Informações sobre a pontuação

- Em um conjunto de casos de teste somando 40 pontos, $N \leq 1000$

Exemplos

Entrada	**Saída**
10 20 5 10 5 15 15 15 5 6 22	5

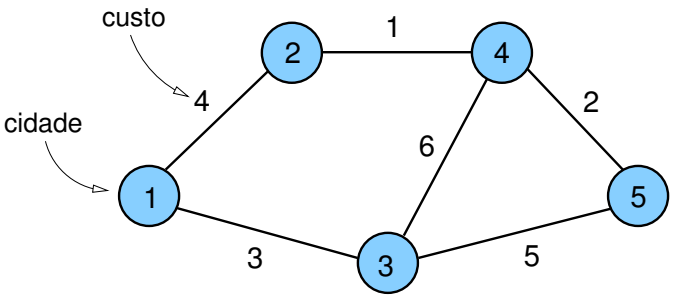
Entrada	**Saída**
5 10 20 30 40 50	2

Frete

Nome do arquivo: `frete.c`, `frete.cpp`, `frete.pas`, `frete.java`, `frete.js`, `frete.py2` *ou* `frete.py3`

O senhor Satoshi passou anos reclamando da empresa de correios do seu país, porque ela sempre transportava suas encomendas usando um caminho que passava pelo número mínimo de cidades entre a cidade onde o senhor Satoshi mora e a cidade destino da encomenda. A empresa alegava que essa estratégia levava ao menor tempo para a entrega final da encomenda. O problema é que ele notou que essa estratégia da empresa nem sempre levava ao menor preço para o frete total. Se ele pudesse escolher o caminho por onde a encomenda deveria passar para ir da sua cidade para a cidade destino, ele poderia economizar bastante com o frete, já que não havia muita urgência para a maioria de suas encomendas.

Depois de muita reclamação, a empresa finalmente está dando aos clientes a opção de determinar o caminho por onde a encomenda deve passar. O senhor Satoshi, feliz da vida, agora quer a sua ajuda para implementar um programa que, dado o custo de transporte de uma encomenda entre vários pares de cidades pelo país, para os quais a empresa realiza entregas diretas, determine qual é o preço total mínimo para o frete entre a cidade onde ele mora e a cidade destino da encomenda.



O país tem N cidades, identificadas pelos números de 1 a N. O senhor Satoshi mora na cidade 1 e o destino da encomenda será sempre a cidade N. É garantido que sempre haverá um caminho de 1 até N. No exemplo da figura, para $N = 5$, o custo mínimo será 7, para o caminho $1 \to 2 \to 4 \to 5$.

Entrada

A primeira linha da entrada contém dois números inteiros N e M, representando o número de cidades e quantos pares de cidades possuem entrega direta de encomenda pela empresa. As M linhas seguintes contêm, cada uma, três inteiros A, B e C, indicando que a empresa realiza a entrega de uma encomenda diretamente entre as cidades A e B, cobrando o preço C.

Saída

Seu programa deve imprimir uma linha contendo um inteiro representando o preço mínimo total para o frete entre a cidade onde o senhor Satoshi mora, a cidade 1, e a cidade destino da encomenda, a cidade N.

Restrições

- $2 \leq N \leq 100$ e $1 \leq M \leq 1000$
- $1 \leq A, B \leq N$ e $A \neq B$
- $1 \leq C \leq 1000$

Exemplos

Entrada	**Saída**
5 6 1 2 4 1 3 3 4 3 6 4 5 2 2 4 1 3 5 5	7

Entrada	**Saída**
7 10 1 2 5 3 1 32 1 4 3 2 3 4 2 6 20 6 3 1 6 4 9 6 5 6 3 7 18 5 7 2	18