OBI2009

Caderno de Tarefas

Modalidade **Programação** • Nível **2**, Fase **1**

A PROVA TEM DURAÇÃO DE **5** HORAS

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 8 páginas (não contando esta 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.
- 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 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*. Para problemas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada problema.
- Ao final da prova, para cada solução que você queira submeter para correção, copie o arquivo fonte para o seu diretório de trabalho ou disquete, 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*.
- Procure resolver o problema 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.

Promoção:

Sociedade Brasileira de Computação

Patrocínio:

Fundação Carlos Chagas

Notas da Prova

Nome do arquivo fonte: `nota.c`, `nota.cpp`, *ou* `nota.pas`

Rosy é uma talentosa professora do Ensino Médio que já ganhou muitos prêmios pela qualidade de sua aula. Seu reconhecimento foi tamanho que foi convidada a dar aulas em uma escola da Inglaterra. Mesmo falando bem inglês, Rosy ficou um pouco apreensiva com a responsabilidade, mas resolveu aceitar a proposta e encará-la como um bom desafio.

Tudo ocorreu bem para Rosy até o dia da prova. Acostumada a dar notas de 0 (zero) a 100 (cem), ela fez o mesmo na primeira prova dos alunos da Inglaterra. No entanto, os alunos acharam estranho, pois na Inglaterra o sistema de notas é diferente: as notas devem ser dadas como conceitos de A a E. O conceito A é o mais alto, enquanto o conceito E é o mais baixo.

Conversando com outros professores, ela recebeu a sugestão de utilizar a seguinte tabela, relacionando as notas numéricas com as notas de conceitos:

Nota	Conceito
0	E
1 a 35	D
36 a 60	C
61 a 85	B
86 a 100	A

O problema é que Rosy já deu as notas no sistema numérico, e terá que converter as notas para o sistema de letras. Porém, Rosy precisa preparar as próximas aulas (para manter a qualidade que a tornou reconhecida), e não tem tempo suficiente para fazer a conversão das notas manualmente.

Tarefa

Você deve escrever um programa que recebe uma nota no sistema numérico e determina o conceito correspondente.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). A entrada contém uma única linha com um número inteiro N ($0 \leq N \leq 100$), representando uma nota de prova no sistema numérico.

Saída

Seu programa deve imprimir, na *saída padrão*, uma letra (A, B, C, D, ou E em maiúsculas) representando o conceito correspondente à nota dada na entrada.

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 35 pontos, $N \leq 10$.
- Em um conjunto de casos de teste que totaliza 70 pontos, $N \leq 50$.

Exemplos

Entrada	**Saída**
12	D

Entrada	**Saída**
87	A

Entrada	**Saída**
0	E

Caçadores de Mitos

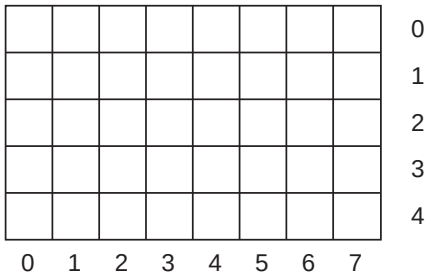
Nome do arquivo fonte: `mito.c`, `mito.cpp`, *ou* `mito.pas`

Jorge é um apresentador de televisão que comanda a versão brasileira do grande sucesso Caçadores de Mitos, onde se estuda um mito para descobrir se é fato ou apenas um boato.

No próximo episódio, Jorge deverá apresentar o mito que diz que "os raios não caem duas vezes no mesmo lugar", referindo-se aos raios das tempestades de chuva.

Para isso, foi até a cidade de Eletrolândia, que é a cidade com maior ocorrência de raios no mundo. O prefeito tem tanto orgulho desse título que mandou criar um sistema para registrar os raios. Jorge conseguiu um relatório com as ocorrências de cada raio que caiu na cidade nos últimos anos.

O mapa de Eletrolândia é um retângulo. Para o sistema de registro a cidade é subdividida em quadrados de um metro de lado, denominados quadrantes. Assim, se a cidade tem 300 metros de largura e 1000 de comprimento, ela será subdividida em 300.000 quadrantes. O sistema de registro armazena o quadrante em que o raio caiu. Cada quadrante é identificado pelas suas coordenadas X e Y, conforme ilustra a figura abaixo, que exemplifica um mapa de uma cidade com oito metros de comprimento por cinco metros de largura (quarenta quadrantes).



Como os quadrantes são relativamente pequenos, Jorge decidiu que se dois raios caíram no mesmo quadrante, pode-se considerar que caíram no mesmo lugar.

Tarefa

Sua missão é escrever um programa que receba as coordenadas dos raios que caíram em Eletrolândia nos últimos anos e determine se o mito estudado é realmente apenas um mito ou pode ser considerado verdade.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado).

A primeira linha da entrada contém um número inteiro N ($2 \leq N \leq 500.000$) representando o número de registros de raios no relatório. Cada uma das N linhas seguintes contém 2 números inteiros X, Y ($0 \leq X, Y \leq 500$), representando o registro de um raio que caiu no quadrante cujas coordenadas são (X, Y).

Saída

Seu programa deve imprimir, na *saída padrão*, o número 0 se nenhum raio caiu no mesmo lugar, ou o número 1 caso contrário. Note que você deve imprimir o número 1 mesmo que haja mais do que 1 par de raios que caíram no mesmo lugar.

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $N \leq 1.000$.
- Em um conjunto de casos de teste que totaliza 70 pontos, $N \leq 100.000$.

Exemplos

Entrada	**Saída**
5 1 1 2 3 3 3 4 2 4 4	0

Entrada	**Saída**
8 1 1 2 2 2 3 4 4 2 3 6 5 9 11 10 10	1

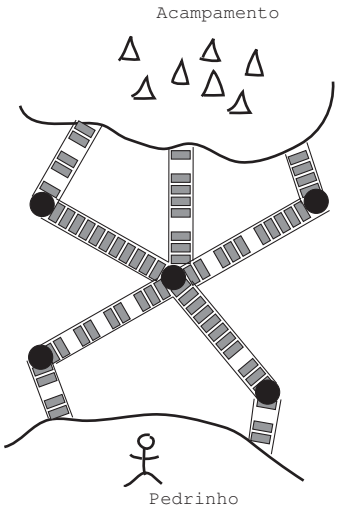
Caminho das Pontes

Nome do arquivo fonte: `pontes.c`, `pontes.cpp`, *ou* `pontes.pas`

Pedrinho é um rapaz muito aventureiro, que nas férias viaja pelo mundo em busca de lugares afastados e com bonitas vistas.

Na sua viagem atual, Pedrinho está andando por uma escura floresta quando se depara com um perigoso desfiladeiro. Do outro lado do desfiladeiro ele sabe que existe um acampamento onde poderá descansar durante a noite para continuar suas aventuras no dia seguinte.

Para chegar até o acampamento, ele terá que utilizar pontes que estão suspensas sobre o desfiladeiro. As pontes foram construídas interligando altos pilares cravados no fundo do desfiladeiro.



O piso das pontes é feita de tábuas de tamanhos iguais. Mas as pontes são velhas, e algumas tábuas caíram. Felizmente, todas as tábuas que sobraram estão em perfeitas condições, ou seja, não existe o perigo de Pedrinho pisar em uma delas e a tábua cair. Além disso, em nenhuma das pontes duas tábuas consecutivas caíram, de forma que os buracos deixados pelas tábuas que caíram podem ser pulados com segurança.

No local onde Pedrinho se encontra existe uma placa mostrando as ligações entre as pontes e também quantas tábuas estão faltando em cada uma das pontes. Pedrinho está cansado e não há muita visibilidade durante a noite. Ele precisa, portanto, tomar muito cuidado para não cair em algum dos buracos.

Pedrinho possui um laptop na mochila, mas só o usa para comunicar-se com os amigos. Ele liga sua internet via satélite, encontra você on-line, e pede sua ajuda.

Tarefa

Sua tarefa é escrever um programa que receba as informações sobre as pontes (as ligações entre elas e a quantidade de tábuas faltando em cada uma) e calcule qual é o menor número de buracos que Pedrinho precisa pular para chegar ao outro lado do desfiladeiro.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado).

A primeira linha da entrada contém dois número inteiros N e M ($1 \leq N \leq 1.000$, $2 \leq M \leq 10.000$) representando o número de pilares no desfiladeiro e o número de pontes, respectivamente. Cada uma das M linhas seguintes

contém 3 inteiros S, T, B ($0 \leq S \leq N+1, 0 \leq T \leq N+1$ e $0 \leq B \leq 1.000$), indicando que existe uma ponte ligando os pilares S e T, e que possui B buracos. Não existe linha representando ponte com $S = T$. O valor de pilar 0 representa a borda do desfiladeiro onde Pedrinho está, e o valor de pilar $N+1$ representa a borda do desfiladeiro onde está o acampamento. Não existem duas pontes distintas ligando o mesmo par de locais (pilares ou bordas do desfiladeiro).

Você pode supor que sempre existirá um caminho de pontes entre o lado do desfiladeiro em que Pedrinho se encontra até o lado do desfiladeiro onde está o acampamento.

Saída

Seu programa deve imprimir, na *saída padrão*, um número inteiro representando a menor quantidade de buracos que Pedrinho terá que pular para conseguir chegar ao acampamento.

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $N \leq 100$ e $M \leq 100$.
- Em um conjunto de casos de teste que totaliza 70 pontos, $N \leq 500$ e $M \leq 5.000$.

Exemplos

Entrada	**Saída**
2 5 0 1 1 0 2 3 0 3 9 1 3 2 2 3 2	3

Entrada	**Saída**
4 9 0 1 1 0 3 4 0 4 2 1 2 5 1 5 3 2 5 5 3 4 2 3 5 5 4 5 8	4

O Fugitivo

Nome do arquivo fonte: `fugitivo.c`, `fugitivo.cpp`, *ou* `fugitivo.pas`

Demasi é um terrorista e mafioso italiano que tentou escapar vindo para o Brasil. Mas Demasi não contava com a astúcia de nossa polícia, e acabou sendo preso aqui também.

Por ser mafioso, Demasi conseguiu contratar advogados muito bons, que através de muitos recursos na justiça, acabaram conseguindo uma liberdade condicional para ele.

Nessa liberdade condicional, Demasi deve permanecer a uma certa distância da delegacia de polícia responsável por ele. Para monitorá-lo melhor, eles instalaram nele uma coleira eletrônica inquebrável que, minuto a minuto, envia para uma central as movimentações de Demasi naquele momento.

A informação da coleira é enviada indicando uma direção e uma distância. Por exemplo, em quatro minutos chegam as quatro linhas de informação abaixo:

```
N 30
O 44
S 22
L 10
```

Isso significa que no primeiro minuto Demasi se deslocou 30 metros para o norte (letra N), no minuto seguinte andou 44 metros para o oeste (letra O), no outro minuto andou 22 metros para o sul (letra S) e no quarto minuto se deslocou 10 metros para o leste (letra L). Para poder dar um castigo ao terrorista, o juiz decidiu que Demasi só poderia andar nas quatro direções citadas acima. Ou seja, Demasi nunca se movimenta na direção noroeste, por exemplo. Neste problema, você pode supor que todos os movimentos de Demasi ocorrem sobre um plano cartesiano.

A polícia precisa estar sempre atenta à movimentação dele, e pede a sua ajuda para verificar se em algum momento o italiano se desloca a uma distância da delegacia maior do que a permitida. A distância considerada para esta medida é a distância euclidiana.

Tarefa

Sua missão é criar um programa que receba as informações da coleira de Demasi e diga se em algum momento Demasi esteve a uma distância maior do que a permitida.

Você deve assumir que no instante 0 (zero) Demasi está dentro da delegacia (ou seja, a uma distância zero).

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado).

A primeira linha da entrada contém dois inteiros N e M ($2 \leq N \leq 500.000$, $1 \leq M \leq 1.000.000$) representando o número de registros enviados pela coleira de Demasi e a distância máxima que ele pode ficar da delegacia, respectivamente. As N linhas seguintes contêm os registros da coleira, em ordem de envio. Cada linha contém um caractere C ('N', 'S', 'L' ou 'O', como especificados acima) e um inteiro D ($1 \leq D \leq 1.000$) representando a distância percorrida no minuto.

Saída

Seu programa deve imprimir, na *saída padrão*, o valor 1 se em algum momento Demasi se afastou da delegacia além da distância permitida, ou o valor 0 caso contrário.

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $N \leq 1.000$, $M \leq 10.000$ e $D \leq 30$.
- Em um conjunto de casos de teste que totaliza 70 pontos, $N \leq 200.000$ e $M \leq 400.000$.

Exemplos

Entrada	**Saída**
5 10 N 2 L 3 S 4 O 4 O 3	0

Entrada	**Saída**
5 10 N 6 L 8 S 15 O 5 O 4	1