OBI2012

Caderno de Tarefas

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

12 de maio de 2012

A PROVA TEM DURAÇÃO DE **5** HORAS

Promoção:

Sociedade Brasileira de Computação

Patrocínio:

Fundação Carlos Chagas

v1.0

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 9 páginas (não contando a folha de rosto), numeradas de 1 a 9. 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*; soluções na linguagem Java devem ser arquivos com sufixo *.java* e a classe principal deve ter o mesmo nome do arquivo fonte; e soluções na linguagem Python devem ser arquivos com sufixo *.py*. 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*.
  - em Java: qualquer classe ou função padrão, como por exemplo *Scanner, BufferedReader, BufferedWriter* e *System.out.println*
  - em Python: *read,readline,readlines,print,write*
- 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.

Álbum de fotos

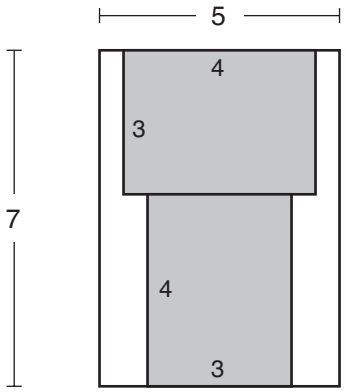
Nome do arquivo fonte: `album.c`, `album.cpp`, `album.pas`, `album.java`, *ou* `album.py`

Clara está organizando as fotos da sua última viagem num álbum de fotos. Como ela tem muitas fotos, para economizar páginas do álbum ela quer colar duas fotos por página do álbum.

Como as fotos são retangulares, as fotos podem ser coladas giradas (mas sempre com lados paralelos aos da página do álbum, para preservar o equilíbrio estético do álbum), mas elas devem sempre ficar inteiramente contidas no interior da página, e não devem se sobrepor.

Em geral, das muitas formas de posicionar as fotos do álbum só algumas (ou nenhuma) satisfazem estas restrições, então pode ser difícil decidir se é possível colar as duas fotos em uma mesma página do álbum, e por isso Clara pediu a sua ajuda para escrever um programa que, dadas as dimensões da página e das fotos, decide se é possível colar as fotos na página.

Por exemplo, cada página pode ser 5×7, e duas fotos são 3×4. Nesse caso, é possível colar as duas fotos:



Entrada

A primeira linha da entrada contém dois inteiros X e Y, indicando a largura e a altura da página do álbum. Cada uma das duas linhas seguintes contém dois inteiros L e H, indicando a largura e a altura das fotos.

Saída

Imprima uma única linha, contendo um único caractere: '`S`', se é possível colar as duas fotos na página do álbum, e '`N`', caso contrário.

Restrições

- $1 \leq X, Y \leq 1000$
- $1 \leq L, H \leq 1000$

Exemplos

Entrada	**Saída**
7 5 3 4 3 4	S

Entrada	**Saída**
10 10 6 6 6 6	N

Entrada	**Saída**
13 8 4 9 6 5	N

Soma das casas

Nome do arquivo fonte: `soma.c`, `soma.cpp`, `soma.pas`, `soma.java`, *ou* `soma.py`

Joãozinho mora em uma rua que tem N casas. Marquinhos é o melhor amigo dele, mas sempre gosta de pregar peças em Joãozinho. Desta vez, ele pegou os dois brinquedos prediletos de Joãozinho e os escondeu em duas casas distintas da rua. Em compensação, Marquinhos deu uma dica importante para Joãozinho:

A soma dos números das casas em que escondi teus brinquedos é igual a K. Além disso, escolhi as casas de tal forma que não existe outro par de casas cuja soma tenha esse mesmo valor.

Sabendo disto, encontre qual é o par de casas em que se encontram os brinquedos de Joãozinho. Para auxiliar seu amigo, Marquinhos entregou a Joãozinho uma lista com o número das casas já em ordem crescente (isto é, do menor para o maior número).

Entrada

A primeira primeira linha da entrada contém um número inteiro N, que representa o número de casas que existem na rua. Cada uma das N linhas seguintes contém um número inteiro, representando o número de uma casa. Note que esses N números estão ordenados, do menor para o maior. A última linha da entrada contém um inteiro K, que é a soma dos números das duas casas onde os brinquedos estão escondidos.

Saída

Se programa deve imprimir uma única linha, contendo dois inteiros, A e B, $A < B$, que representam os números das casas em que estão escondidos os brinquedos. Os dois números devem ser separados por um espaço em branco.

Informações sobre a pontuação

- Em um conjunto de casos que totaliza 30 pontos, $N \leq 10^3$.

Restrições

- $2 \leq N \leq 10^5$
- Para cada casa C_i, $0 \leq C_i \leq 10^9$, $i = 1, 2, \ldots, N$
- Os números das casas estão em ordem crescente, do menor para o maior número, e casas distintas têm números distintos.

Exemplos

Entrada	**Saída**
4 1 2 3 5 8	3 5

Entrada	**Saída**
4 1 2 3 5 5	2 3

Bomba

Nome do arquivo fonte: `bomba.c`, `bomba.cpp`, `bomba.pas`, `bomba.java`, *ou* `bomba.py`

Um terrorista internacional telefonou avisando que há uma bomba a bordo de um dos diversos ônibus interestaduais da Nlogônia. Essa bomba explodirá se, por qualquer motivo, o ônibus for obrigado a parar. O esquadrão anti-bombas já se posicionou na estrada para desarmar a bomba em movimento, mas o ônibus está prestes a entrar na capital da Nlogônia, Nlogópolis, e precisa sair de lá para o esquadrão poder desarmar o artefato. Por questões de segurança, o esquadrão anti-bombas somente pode desarmar o artefato fora da capital.

No projeto urbano de Nlogópolis, todas as interseções consistem de rotatórias, de forma que os veículos nunca precisam parar nas interseções. Em compensação, toda rua (que tem mão única e sempre liga duas rotatórias) possui uma faixa de pedestres com um semáforo; enquanto alguns semáforos abrem nos minutos múltiplos de 3 e fecham nos demais, outros fecham nos minutos múltiplos de 3 e abrem nos demais. Todas as ruas de Nlogópolis foram projetadas de tal forma que sempre levam exatamente um minuto para serem percorridas.

O ônibus vai entrar em Nlogópolis exatamente meio-dia em ponto em uma das rotatórias, e deve sair por outra rotatória específica para encontrar o esquadrão anti-bombas na estrada. O comandante da polícia local lhe pediu que escreva um programa que determina o menor tempo necessário para que o ônibus saia da cidade, pela rotatória específica de saída. Note que o ônibus pode ser forçado a parar em um semáforo, por falta de alternativas adequadas, e nesse caso a bomba explodirá. Ele também pode ficar circulando indefinidamente pela cidade, e nesse caso eventualmente terá que parar por falta de combustível (e a bomba explodirá).

Entrada

A primeira linha da entrada contém quatro inteiros N, E, S, M, indicando, respectivamente, o número de rotatórias (numeradas de 0 a $N-1$), o número da rotatória de entrada do ônibus, o número da rotatória de saída do ônibus e o número de ruas da cidade.

Cada uma das M linhas seguintes contém três inteiros A, B e T, indicando respectivamente a rotatória de origem da rua, a rotatória de destino da rua e a temporização do semáforo daquela rua: $T = 1$ se o semáforo daquela rua abre nos minutos múltiplos de 3, e $T = 0$ se o semáforo daquela rua fecha nos minutos múltiplos de 3.

Saída

Imprima uma única linha contendo um único número inteiro, o menor tempo necessário em minutos para que o ônibus saia da cidade ileso. Se for impossível evitar a explosão do ônibus, imprima uma única linha contendo o caractere '`*`'.

Restrições

- $2 \leq N \leq 500$
- $1 \leq M \leq 2000$
- $0 \leq E, S \leq N-1$
- pode haver até duas ruas de uma rotatória A para outra B (possivelmente igual a A), mas no caso de haver duas ruas, então numa o semáforo abre nos minutos múltiplos de 3, na outra o semáforo fecha nos minutos múltiplos de 3.

Exemplos

Entrada	**Saída**
6 5 4 7 0 1 0 1 2 0 1 2 1 2 3 1 2 4 0 3 0 0 5 0 1	8

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

Banco

Nome do arquivo fonte: `banco.c`, `banco.cpp`, `banco.pas`, `banco.java`, *ou* `banco.py`

A legislação em vigor obriga os bancos a iniciarem o atendimento a um cliente em no máximo 20 minutos após a entrada do cliente na fila única da agência bancária. A fila é única, assim um caixa livre solicita ao primeiro cliente da fila que venha ao seu guichê para ser atendido. (Vamos ignorar aqui o problema dos clientes prioritários, idosos, gestantes, portadores de necessidades especiais, etc.) Estamos supondo também que nenhum caixa atende dois clientes ao mesmo tempo.

Seu programa receberá o número de caixas ativas na agência, o número de clientes e, para cada cliente, duas informações, a saber, o momento de entrada do cliente na fila, e a duração do atendimento daquele cliente.

Inicialmente todos os caixas estão vazios, já que a agência acabou de abrir.

Seu problema é determinar o número de clientes que esperarão mais de 20 minutos para ter seu atendimento iniciado.

Entrada

A primeira linha da entrada contém dois inteiros separados por um espaço em branco. O primeiro, C, é o número de caixas ativas na agência bancária. O segundo, N, o número de clientes que procurarão atendimento na agência naquele dia.

As próximas N linhas terão cada uma informações sobre um cliente, consistindo de dois inteiros, T e D, separados por um espaço em branco. O inteiro T fornece o momento em que o cliente entra na fila, em minutos, a partir do instante de abertura da agência. O inteiro D fornece, em minutos, o tempo necessário para atender o cliente.

As linhas estão ordenadas por entrada dos clientes na fila.

Saída

A saída deverá conter apenas uma linha, contendo um único inteiro, o número de clientes cujo atendimento será iniciado mais do que 20 minutos após sua entrada na fila.

Restrições

- $1 \leq C \leq 10$
- $1 \leq N \leq 1000$
- $0 \leq T \leq 300$
- $1 \leq D \leq 10$

Exemplos

Entrada	**Saída**
1 5 0 10 0 10 1 10 2 10 30 10	1

Entrada	**Saída**
3 16 0 10 0 10 0 10 3 10 5 10 7 10 11 10 13 10 14 10 15 10 16 10 17 10 18 3 19 10 20 10 23 3	2