Competidor(a): _____

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

OBI2020

Caderno de Tarefas

Modalidade **Programação** • **Nível Júnior** • Fase **Nacional**

28 de novembro de 2020

A PROVA TEM DURAÇÃO DE **3** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

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.

Aplicativo de Calorias

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

Um aplicativo de celular está sendo desenvolvido para, a partir da foto de um prato contendo uma refeição, estimar a quantidade de calorias da refeição.

O algoritmo de inteligência artificial (IA) utilizado no aplicativo produz três números inteiros, E_1, E_2 e E_3. E_1 é a quantidade mínima de calorias estimada e E_2 a quantidade máxima de calorias estimada para a refeição da fotografia. E_3 só tem significado se a diferença entre as quantidades estimadas mínima e máxima são maiores do que um valor pré-definido X; nesse caso, $E3$ é a quantidade de calorias estimada por um método alternativo.

Depois de vários testes, os desenvolvedores do aplicativo determinaram que os melhores resultados são obtidos usando as estimativas produzidas pelo algoritmo de IA da seguinte forma:

- se a diferença entre E_1 e E_2 for menor ou igual ao valor de X, o aplicativo deve mostrar ao usuário o valor de E_2 como o número de calorias;
- se a diferença entre E_1 e E_2 for maior do que o valor de X, o aplicativo deve mostrar ao usuário o valor de E_3 como o número de calorias;

Dados o valor de X e as três estimativas produzidas pelo algoritmo de IA, escreva um programa que determine o resultado que deve ser mostrado para o usuário.

Entrada

A primeira linha da entrada contém um inteiro, o valor de E_1. A segunda linha contém um inteiro, o valor de E_2. A terceira linha contém um inteiro, o valor de E_3. A quarta linha contém um inteiro, o valor de X.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o resultado que deve ser mostrado para o usuário do aplicativo.

Restrições

- $0 \leq E_1 \leq E_2 \leq 10000$
- $0 \leq E_3 \leq 10000$
- $0 \leq X \leq 10000$

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
1500 2000 2500 1000	2000

Exemplo de entrada 2	**Exemplo de saída 2**
1000 1300 1050 200	1050

Cobertura para Celular

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

Para atrair mais turistas, o governo decidiu permitir a instalação de uma rede de telefonia celular no paradisíaco arquipélago de Logarium. O arquipélago tem muitas ilhas no formato circular, todas com no máximo 1 km de diâmetro.

Exatamente uma torre de celular será instalada no centro de cada uma das ilhas. Todas as torres serão idênticas e terão o mesmo alcance; o *alcance* é a distância máxima da torre que um equipamento (telefone ou outra torre) pode estar de forma que a comunição seja possível.

O governo deseja que a rede de telefonia celular garanta a *cobertura total* do arquipélago, ou seja, deve ser possível a um usuário comunicar-se com qualquer outro usuário no arquipélago, mesmo que a comunicação tenha que passar por mais de uma torre.

Há vários tipos de torres disponíveis no mercado, cada tipo com um alcance. O governo recebeu uma proposta atrativa de uma empresa e deseja saber se o alcance da torre ofertada permitirá a cobertura total do arquipélago.

Dadas a localização das torres e o alcance da torre ofertada, escreva um programa para determinar se a torre ofertada permite a cobertura total do arquipélago.

Entrada

A primeira linha da entrada contém um inteiro N indicando o número de ilhas do arquipélago. Cada uma das N linhas seguintes contém dois inteiros X_i e Y_i, as coordenadas da i-ésima torre. Não existem duas torres com as mesmas coordenadas. A última linha da entrada contém um inteiro A indicando o alcance da torre.

Saída

Seu programa deve produzir uma única linha na saída, contendo um único caractere, que deve ser `S` se a torre permite a cobertura total ou `N` caso contrário.

Restrições

- $2 \leq N \leq 10000$
- $0 \leq X_i, Y_i \leq 1000$, para $1 \leq i \leq N$
- $1 \leq A \leq 10000$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 20 pontos, $Y_i = 0$ para $1 \leq i \leq N$.
- Para um conjunto de casos de testes valendo 80 pontos adicionais, nenhuma outra restrição.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
3 200 200 400 400 600 600 200	N

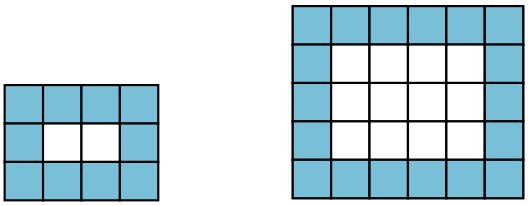
Exemplo de entrada 2	**Exemplo de saída 2**
5 10 10 10 30 30 10 30 30 20 20 20	S

Exemplo de entrada 3	**Exemplo de saída 3**
3 0 0 40 0 0 30 49	S

Atlanta

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

Documentos recentemente encontrados por pesquisadores mostram que na Sala de Audiências do palácio Real na cidade perdida de Atlanta o piso era formado por ladrilhos 20 cm x 20 cm. Ladrilhos de duas cores foram usados: o centro da Sala era formado por ladrilhos brancos e exatamente uma fileira de ladrilhos azuis foram colocados em cada lateral do Sala, como nas figuras abaixo.



Os pesquisadores não encontraram vestígios da Sala de Audiências (nem da cidade de Atlanta!), mas os documentos recentes, se forem autênticos, indicam também a quantidade de ladrilhos que foram utilizados no piso da Sala.

Sua tarefa é, dadas as quantidades de azulejos azuis e brancos, determinar as dimensões da Sala de Audiências.

Entrada

A primeira linha da entrada contém um inteiro A, o número de azulejos azuis. A segunda linha contém um número inteiro B, o número de azulejos brancos.

Saída

Seu programa deve produzir uma única linha, contendo dois números inteiros, representando as dimensões da Sala (largura e comprimento). Se a largura for diferente do comprimento, seu programa deve imprimir primeiro a menor dimensão, seguida da maior dimensão. Se as quantidades de azulejos não forem corretas para construir o piso da Sala no formato descrito acima, seu programa deve imprimir `-1 -1`.

Restrições

- $1 \leq A \leq 10^6$
- $1 \leq B \leq 10^6$

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
`10` `2`	`3 4`

Exemplo de entrada 2	**Exemplo de saída 2**
`8` `2`	`-1 -1`

Exemplo de entrada 3	**Exemplo de saída 3**
3996 996004	1000 1000

Torre de Dados

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

Hortência está brincando de construir uma torre com dados de seis faces. Os dados são similares aos dados comuns utilizados em jogos, com as faces estampadas com valores de 1 a 6. Mas os dados usados por Hortência têm uma grande diferença em relação aos dados comuns: enquanto em dados comuns a soma dos valores em faces opostas é sempre sete, para os dados de Hortência os valores em faces opostas nem sempre têm soma sete.

Hortência está criando a torre empilhando os seus dados, obedecendo às seguintes regras:

1. Seja X dado que está colocado imediatamente em cima de um dado Y. Então o valor da face inferior de X deve ser igual ao valor da face superior de Y. Por exemplo, se a pilha tem três dados R, S e T, empilhados nessa ordem de baixo para cima, então o valor da face superior de R deve ser igual ao valor da face inferior de S, e o valor da face superior de S deve ser igual ao valor da face inferior de T.

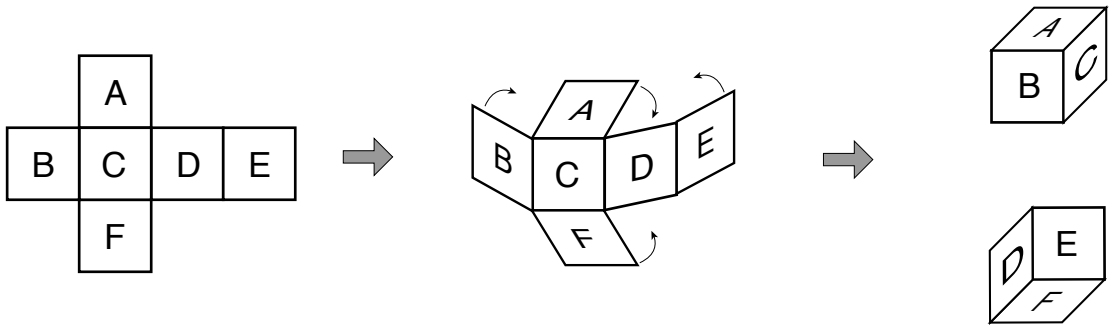
2. Os dados devem ter suas laterais alinhadas, ou seja, a torre formada tem exatamente quatro lados, correspondendo aos lados dos dados.

Hortência quer criar uma torre tal que a soma dos valores de um dos lados da torre seja a maior possível. Note que após empilhar os dados obedecendo à regra (1), Hortência pode girar cada dado horizontalmente de forma independente, obedecendo à regra (2), para alterar os valores das somas dos lados.

Dadas as informações sobre os dados utilizados na torre, escreva um programa para determinar o maior valor possível para a soma de um dos lados da torre.

Entrada

A primeira linha da entrada contém um inteiro N, o número de dados. Cada uma das N linhas seguintes descreve um dado e contém seis inteiros A, B, C, D, E e F, identificando os valores dos lados do dado, conforme a figura abaixo. *Errata, faltou a seguinte frase na prova online: os dados são empilhados na torre na ordem dada na entrada, de baixo para cima. As submisões serão julgadas com dois conjuntos de testes, um considerando os dados na ordem dada na entrada e outro considerando que a ordem pode ser qualquer uma e o competidor receberá a maior pontuação entre as duas correções.*



Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o maior valor possível para a soma de um dos lados da torre.

Restrições

- $1 \leq N \leq 1000$
- $1 \leq A, B, C, D, E, F \leq 6$, todos distintos em um dado.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
2 1 2 3 4 5 6 1 2 6 4 5 3	12

Exemplo de entrada 2	**Exemplo de saída 2**
5 4 6 1 5 3 2 1 3 2 4 6 5 5 3 1 4 2 6 6 1 5 2 4 3 4 6 1 5 3 2	28