OBI2017

Caderno de Tarefas

Modalidade **Programação** • **Nível Júnior** • Fase **1**

12 de maio de 2017

A PROVA TEM DURAÇÃO DE **2** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

v1.1

Instruções

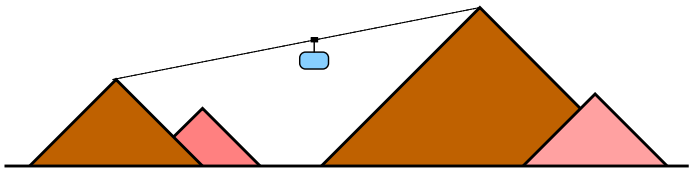
LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 4 páginas (não contando a folha de rosto), numeradas de 1 a 4. 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; soluções na linguagem Python devem ser arquivos com sufixo *.py2* para python2 e .py3 para python3; e soluções na linguagem Javascript devem ter arquivos com sufixo *.js*. 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 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 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.

Bondinho

Nome do arquivo: `bondinho.c`, `bondinho.cpp`, `bondinho.pas`, `bondinho.java`, `bondinho.js` *ou* `bondinho.py`

A turma do colégio vai fazer uma excursão na serra e todos os alunos e monitores vão tomar um bondinho para subir até o pico de uma montanha. A cabine do bondinho pode levar 50 pessoas no máximo, contando alunos e monitores, durante uma viagem até o pico. Neste problema, dado como entrada o número de alunos A e o número de monitores M, você deve escrever um programa que diga se é possível ou não levar todos os alunos e monitores em apenas uma viagem!



Entrada

A primeira linha da entrada contém um inteiro A, representando a quantidade de alunos. A segunda linha da entrada contém um inteiro M, representando o número de monitores.

Saída

Seu programa deve imprimir uma linha contendo o caractere `S` se é possível levar todos os alunos e monitores em apenas uma viagem, ou o caractere `N` caso não seja possível.

Restrições

- $1 \leq A \leq 50$
- $1 \leq M \leq 50$

Exemplos

Entrada	**Saída**
10 20	S

Entrada	**Saída**
12 39	N

Entrada	**Saída**
49 1	S

Drone de Entrega

Nome do arquivo: `drone.c`, `drone.cpp`, `drone.pas`, `drone.java`, `drone.js` *ou* `drone.py`

A loja do Pará, especializada em vendas pela internet, está desenvolvendo *drones* para entrega de caixas com as compras dos clientes. Cada caixa tem a forma de um paralelepípedo reto retângulo (ou seja, no formato de um tijolo).

O *drone* entregará uma caixa de cada vez, e colocará a caixa diretamente dentro da casa do cliente, através de uma janela. Todas as janelas dos clientes têm o formato retangular e estão sempre totalmente abertas. O *drone* tem um aplicativo de visão computacional que calcula exatamente as dimensões H e L da janela. O *drone* consegue colocar a caixa através da janela somente quando uma das faces da caixa está paralela à janela, mas consegue virar e rotacionar a caixa antes de passá-la pela janela.

O aplicativo de controle do *drone* está quase pronto, mas falta um pequeno detalhe: um programa que, dadas as dimensões da maior janela do cliente e as dimensões da caixa que deve ser entregue, determine se o *drone* vai ser capaz de entregar a compra (pela janela) ou se a compra terá que ser entregue por meios normais.

Entrada

A entrada é composta por cinco linhas, cada uma contendo um número inteiro. A três primeiras linhas contêm os valores A, B, C, indicando as três dimensões da caixa, em centímetros. As duas últimas linhas contêm os valores H e L, indicando a altura e a largura da janela, em centímetros.

Saída

Seu programa deve escrever uma única linha, contendo apenas a letra `S` se a caixa passa pela janela e apenas a letra `N` em caso contrário.

Restrições

- $1 \leq A, B, C \leq 80$
- $1 \leq H, L \leq 100$

Exemplos

Entrada	**Saída**
30 50 80 80 60	S

Entrada	**Saída**
75 100 50 100 30	N

Entrada	**Saída**
20 22 5 20 10	S