Competidor(a): _____

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

# OBI2020

# Caderno de Tarefas

Modalidade **Programação** • **Nível Júnior** • Fase **Local**

15 e 16 de junho de 2020

A PROVA TEM DURAÇÃO DE **2** HORAS

**Promoção:**

Sociedade Brasileira de Computação

**Apoio:**

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

# Instruções

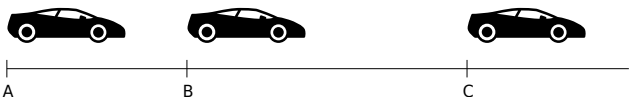
LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 6 páginas (não contando a folha de rosto), numeradas de 1 a 6. 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.

# Piloto Automático

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

Uma grande fábrica de carros elétricos está realizando melhorias no sistema de piloto automático e precisa da sua ajuda para implementar um programa que decida se um carro B, que está trafegando no meio de dois carros A e C, precisa acelerar, desacelerar ou manter a velocidade atual. Os carros são iguais e os sensores do piloto automático vão fornecer, como entrada, a posição atual da traseira dos três carros. Veja um exemplo na figura.




O carro B precisa ser acelerado se a distância da sua traseira para a traseira do carro A for menor do que a distância da sua traseira para a traseira do carro C. Se for maior, ele precisa ser desacelerado. Se for igual, precisa manter a velocidade atual. Quer dizer, o carro B precisa ser acelerado se $(B - A) < (C - B)$, desacelerado se $(B - A) > (C - B)$ e manter a velocidade se $(B - A)$ for igual a $(C - B)$.

## Entrada

A primeira linha da entrada contém um inteiro $A$. A segunda linha da entrada contém um inteiro $B$. A terceira linha da entrada contém um inteiro $C$. Os três inteiros representam as posições atuais das traseiras dos carros A, B e C, respectivamente.

## Saída

Seu programa deve imprimir uma linha contendo um inteiro: 1 se o carro B precisa acelerar; $-1$ se precisa desacelerar; ou 0 se precisa manter a velocidade atual.

## Restrições

- $0 \leq A < B < C \leq 500$

## Exemplos

| **Exemplo de entrada 1** | **Exemplo de saída 1** |
|---|---|
| 10<br>23<br>38 | 1 |

| **Exemplo de entrada 2** | **Exemplo de saída 2** |
|---|---|
| 105<br>212<br>319 | 0 |

| **Exemplo de entrada 3** | **Exemplo de saída 3** |
|---|---|
| 80<br>120<br>132 | -1 |

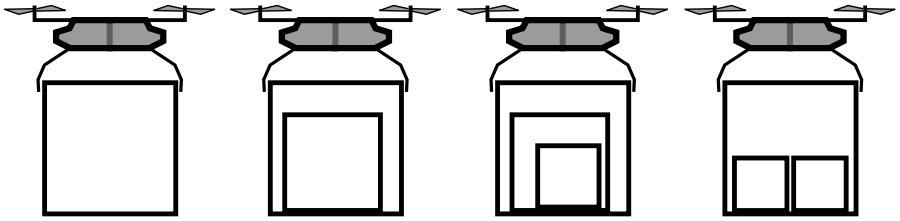
# Entrega de Caixas

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

Você precisa transportar três caixas vazias usando um drone que pode levantar uma caixa por vez apenas em cada viagem. Quer dizer, sempre dá para transportar as três caixas vazias fazendo três viagens do drone. Mas talvez dê para fazer menos do que três viagens, se for possível colocar uma caixa dentro de outra. As caixas têm formato de cubo e a única restrição para uma caixa ser colocada dentro de outra é o tamanho, não importando o peso.

Uma caixa de tamanho $X$ pode ser colocada dentro de uma caixa de tamanho $Y$ se $X < Y$. Note, portanto, que uma caixa não cabe dentro de outra do mesmo tamanho. Além disso, duas caixas de tamanhos $X$ e $Y$ podem ser colocadas, lado a lado, dentro de uma caixa de tamanho $Z$ se $(X + Y) < Z$.

A figura ilustra as quatro configurações possíveis para o drone fazer uma viagem.



Neste problema, os tamanhos das três caixas são dados em ordem crescente e seu programa deve computar o número mínimo de viagens que o drone pode fazer para transportar todas as três caixas.

## Entrada

A primeira linha da entrada contém um inteiro $A$. A segunda linha da entrada contém um inteiro $B$. A terceira linha da entrada contém um inteiro $C$. Os três inteiros representam os tamanhos das três caixas.

## Saída

Seu programa deve imprimir uma linha contendo um inteiro, representando o número mínimo de viagens que o drone pode fazer para transportar todas as três caixas.

## Restrições

- $1 \leq A \leq B \leq C \leq 1000$

## Exemplos

| **Exemplo de entrada 1** | **Exemplo de saída 1** |
|---|---|
| 12<br>45<br>188 | 1 |

| **Exemplo de entrada 2** | **Exemplo de saída 2** |
|---|---|
| 67<br>67<br>67 | 3 |

| **Exemplo de entrada 3** | **Exemplo de saída 3** |
|---|---|
| 111<br>463<br>463 | 2 |

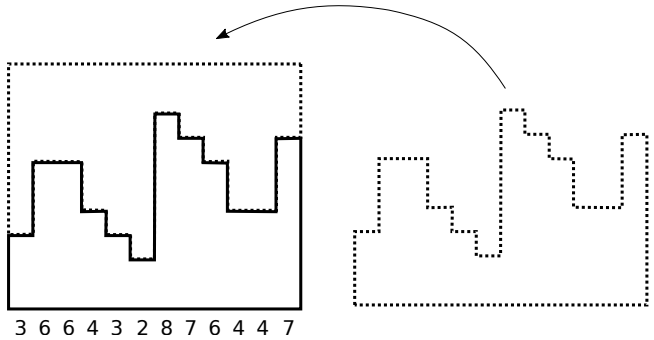
| **Exemplo de entrada 4** | **Exemplo de saída 4** |
|---|---|
| 72<br>72<br>345 | 1 |

| **Exemplo de entrada 5** | **Exemplo de saída 5** |
|---|---|
| 30<br>30<br>55 | 2 |

# Escher

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

M. C. Escher foi um artista gráfico holandês que fazia incríveis ilustrações onde preenchia a tela com objetos auto-similares, cujos contornos encaixam neles próprios, criando simetrias geométricas muito impressionantes. Veja um exemplo dessa ideia na figura, que mostra um objeto que é um perfil ortogonal definido por uma sequência de números naturais representando a sequência de alturas. Podemos pegar uma cópia do objeto, rotacionar 180 graus e encaixar perfeitamente no objeto original, formando um retângulo.




Em termos mais gerais, se uma sequência de $N$ números naturais representando a sequência de alturas for $A_1, A_2, A_3, \ldots, A_{N-2}, A_{N-1}, A_N$, o perfil definido será chamado de perfil Escher se tivermos $A_1 + A_N$ igual a $A_2 + A_{N-1}$ igual a $A_3 + A_{N-2}$, e assim por diante. Neste problema, será dada a sequência de alturas que definem o perfil e seu programa deve decidir se o perfil é Escher, ou não.

## Entrada

A primeira linha da entrada contém um número $N$, indicando quantos números tem a sequência. A segunda linha da entrada contém $N$ números naturais, $A_i$, para $1 \leq i \leq N$, definindo a sequência de alturas do perfil.

## Saída

Seu programa deve imprimir uma linha contendo o caractere `S`, se o perfil for Escher; ou `N`, senão.

## Restrições

- $3 \leq N \leq 10000$.
- $1 \leq A_i \leq 1000$, para todo $1 \leq i \leq N$.

## Informações sobre a pontuação

- Em um conjunto de casos de teste somando 20 pontos, $N = 3$.
- Em um conjunto de casos de teste somando 80 pontos, nenhuma restrição adicional.

## Exemplos

| **Exemplo de entrada 1** | **Exemplo de saída 1** |
|---|---|
| 12<br>3 6 6 4 3 2 8 7 6 4 4 7 | S |

| **Exemplo de entrada 2** | **Exemplo de saída 2** |
|---|---|
| 5<br>2 1 9 13 12 | N |