# OBI2007

# Caderno de Tarefas

Modalidade Programação • Nível 1, Fase 1

17 de Março de 2007

A PROVA TEM DURAÇÃO DE TRÊS HORAS

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 3 páginas (não contando esta folha de rosto), numeradas de 1 a 3. 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.

Sociedade Brasileira de Computação
www.sbc.org.br
Fundação Carlos Chagas
www.fcc.org.br

# Detectando Colisões

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

Detecção de colisão é uma das operações mais comuns (e importantes) em jogos eletrônicos. O objetivo, basicamente, é verificar se dois objetos quaisquer colidiram, ou seja, se a interseção entre eles é diferente de vazio. Isso pode ser usado para saber se duas naves colidiram, se um monstro bateu numa parede, se um personagem pegou um item, etc.

Para facilitar as coisas, muitas vezes os objetos são aproximados por figuras geométricas simples (esferas, paralelepípedos, triângulos etc). Neste problema, os objetos são aproximados por retângulos num plano 2D.

## Tarefa

Escreva um programa que, dados dois retângulos, determine se eles se interceptam ou não.

## Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). Cada caso de teste contém duas linhas. Cada linha contém quatro inteiros ($x_0, y_0, x_1, y_1$, sendo $0 \leq x_0 < x_1 \leq 1.000.000$ e $0 \leq y_0 < y_1 \leq 1.000.000$) separados por um espaço em branco representando um retângulo. Os lados do retângulo são sempre paralelos aos eixos $x$ e $y$.

## Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha para cada caso de teste, contendo o número 0 (zero) caso não haja interseção ou o número 1 (um) caso haja.

| **Entrada** | **Entrada** | **Entrada** |
|---|---|---|
| `0 0 1 1`<br>`0 0 1 1` | `0 0 2 2`<br>`1 1 3 3` | `0 0 1 1`<br>`2 2 3 3` |
| **Saída** | **Saída** | **Saída** |
| `1` | `1` | `0` |

# Peça Perdida

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

Joãozinho adora quebra-cabeças, essa é sua brincadeira favorita. O grande problema, porém, é que às vezes o jogo vem com uma peça faltando. Isso irrita bastante o pobre menino, que tem de descobrir qual peça está faltando e solicitar uma peça de reposição ao fabricante do jogo. Sabendo que o quebra-cabeças tem $N$ peças, numeradas de 1 a $N$ e que exatamente uma está faltando, ajude Joãozinho a saber qual peça ele tem de pedir.

## Tarefa

Escreva um programa que, dado um inteiro $N$ e $N-1$ inteiros numerados de 1 a $N$, descubra qual inteiro está faltando.

## Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). A entrada contém 2 linhas. A primeira linha contém um inteiro $N$ ($2 \leq N \leq 1.000$). A segunda linha contém $N-1$ inteiros numerados de 1 a $N$ (sem repetições).

## Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo o número que está faltando na seqüência dada.

| **Entrada** | **Entrada** | **Entrada** |
|---|---|---|
| `3`<br>`3 1` | `5`<br>`1 2 3 5` | `4`<br>`2 4 3` |
| **Saída** | **Saída** | **Saída** |
| `2` | `4` | `1` |

# Quadrado Mágico

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

Chama-se de *quadrado mágico* um arranjo, na forma de um quadrado, de $N \times N$ números inteiros tal que todas as linhas, colunas e diagonais têm a mesma soma.

Por exemplo, o quadrado abaixo

```
2 7 6
9 5 1
4 3 8
```

é um quadrado mágico de soma 15, pois todas as linhas ($2+7+6=15$, $9+5+1=15$ e $4+3+8=15$), colunas ($2+9+4=15$, $7+5+3=15$ e $6+1+8=15$) e diagonais ($2+5+8=15$ e $6+5+4=15$) têm a mesma soma (15).

## Tarefa

Escreva um programa que, dado um quadrado, determine se ele é magico ou não e qual a soma dele (caso seja mágico).

## 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 de cada caso de teste contém um inteiro $N$ ($2 < N < 10$). As $N$ linhas seguintes contêm $N$ inteiros cada, separados por exatamente um espaço em branco. Os inteiros dentro do quadrado são todos maiores que 0 (zero) e menores que 1.000.

## Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha com um inteiro representando a soma do quadrado mágico ou $-1$ caso o quadrado não seja mágico.

| **Entrada** | **Entrada** | **Entrada** |
|---|---|---|
| `3`<br>`2 7 6`<br>`9 5 1`<br>`4 3 8` | `3`<br>`1 2 3`<br>`4 5 6`<br>`7 8 9` | `4`<br>`16 3 2 13`<br>`5 10 11 8`<br>`9 6 7 12`<br>`4 15 14 1` |
| **Saída** | **Saída** | **Saída** |
| `15` | `-1` | `34` |