



**OBI2011**

## **Caderno de Tarefas**

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

14 de maio de 2011

**A PROVA TEM DURAÇÃO DE 3 HORAS**

**Promoção:**



Sociedade Brasileira de Computação

**Patrocínio:**



Fundação Carlos Chagas

# Instruções

## LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 7 páginas (não contando a folha de rosto), numeradas de 1 a 7. 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.

# Chuva

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

Bob trabalha no OBM (Órgão Brasileiro de Meteorologia), que é a organização responsável pela medição dos índices pluviométricos (quantidade de chuva acumulada) em todo o país. Eles são muito eficientes no que fazem, mas estão com um problema: eles não sabem como proceder para calcular a quantidade acumulada de chuva que caiu em cada região em dois períodos consecutivos, muito embora eles saibam os dados de cada período separadamente.

Como a chefia do Órgão estava muito ocupada, acabou ficando a cargo de Bob, o estagiário, a tarefa de implementar um programa que some, para cada região, a quantidade de chuva acumulada em dois períodos consecutivos.

O mapa que o OBM usa é dividido em  $N \times N$  regiões, sendo que para cada região, a cada período, é determinado um número inteiro indicando a quantidade de chuva acumulada. A quantidade de chuva acumulada total em cada região em dois períodos consecutivos é a soma das quantidades de chuva em cada um dos períodos.

Mas como Bob é só um estagiário e não está acostumado a fazer nada mais do que tirar cópias de documentos, ele pediu sua ajuda para implementar o programa que calcula a quantidade de chuva acumulada total nos dois períodos para cada uma das regiões, dadas as quantidades de chuva acumulada em cada período para cada região.

## Entrada

A primeira linha da entrada contém um inteiro  $N$  indicando a dimensão dos dois mapas que devem ser lidos. Nas próximas  $2N$  linhas são dados os dois mapas, cada mapa indicando a quantidade de chuva acumulada nas regiões em um período. Cada mapa é descrito em  $N$  linhas consecutivas, cada linha contendo  $N$  inteiros, sendo que cada inteiro indica a quantidade de chuva acumulada, no período, em uma região.

## Saída

A saída deverá conter  $N$  linhas, com  $N$  inteiros em cada linha, indicando a quantidade de chuva acumulada total em cada uma das regiões nos dois períodos considerados.

## Restrições

- $1 \leq N \leq 100$ .
- $0 \leq$  quantidade de chuva acumulada em cada região de cada mapa  $\leq 100$ .

## Exemplos

<b>Entrada</b>	<b>Saída</b>
2 1 2 3 4 10 11 12 13	11 13 15 17

<b>Entrada</b>	<b>Saída</b>
3 1 1 1 1 2 2 1 2 3 3 2 1 2 2 1 1 1 1	4 3 2 3 4 3 2 3 4

# Gincana

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

Toda semana Juquinha tem aulas de ACM (Artes Cênicas e Musicais) no colégio em que estuda e, recentemente, sua professora anunciou que haverá uma gincana no final do semestre. No entanto, os times devem ser formados o mais breve possível para que os alunos possam ensaiar.

Cada time é constituído de um ou mais alunos, e cada aluno tem que pertencer a exatamente um time. Além disso, os times não podem ser formados de qualquer maneira: se um aluno é amigo de outro, esses alunos devem estar no mesmo time. A professora então pediu para que os alunos a informassem das relações de amizade na sala de aula.

Os alunos então se numeraram de 1 até  $N$  e escreveram uma lista cujas linhas contém pares de números. Se dois alunos cujos números são  $i$  e  $j$  são amigos, haverá ao menos uma linha contendo  $i$  e  $j$  ou  $j$  e  $i$  na lista. Inversamente, se há uma linha contendo  $i$  e  $j$  na lista, então os alunos cujos números são  $i$  e  $j$  são amigos.

A professora então recolheu a lista e, na próxima aula, deverá decidir que times formar. Ela está pensando em formar o maior número possível de times e gostaria de saber quantos times ela formaria. Ajude então a professora escrevendo um programa que, dada a lista de amizades, determina qual o maior número de times que ela pode formar.

## Entrada

A primeira linha da entrada contém dois inteiros  $N$  e  $M$  que representam, respectivamente, o número de alunos na turma e o número de linhas na lista.

As próximas  $M$  linhas contém a lista de amizades. Cada linha contém dois inteiros  $I$  e  $J$  separados por exatamente um espaço.

## Saída

Seu programa deve imprimir uma linha contendo o número máximo de times que podem ser formados pela professora.

## Restrições

- $1 \leq N \leq 1000$ .
- $0 \leq M \leq 5000$ .
- $1 \leq I, J \leq N$ .

## Exemplos

<b>Entrada</b>	<b>Saída</b>
3 1 1 3	2

<b>Entrada</b>	<b>Saída</b>
7 6 1 6 6 4 5 2 3 7 2 3 7 2	2

# Calculadora

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

Solicitando Boas Contas (SBC) é uma organização de inspeção de calculadoras. Todos os fabricantes procuram ter o selo de qualidade da SBC, que faz com que os clientes comprem o produto sem preocupação com contas erradas.

Você está encarregado de testar máquinas que fazem apenas operações de multiplicação e divisão. Além disso, o termo a ser digitado em cada operação (que dividirá ou multiplicará o número atualmente exibido no visor) só pode conter um único dígito.

A calculadora exibe o número 1 quando ligada. Depois disso, o usuário pode digitar um número com um único dígito e escolher se esse número deve multiplicar ou dividir o número exibido anteriormente; o resultado da operação escolhida é então exibido na calculadora. Pode-se repetir esse processo indefinidamente.

Apesar de só podermos entrar com números inteiros de um dígito, o visor da calculadora permite exibir números com múltiplos dígitos e até mesmo números fracionários.

Dada uma sequência de operações que foram realizadas nessa calculadora logo depois de ligada, sua tarefa é conferir o resultado exibido.

## Entrada

A primeira e única linha da entrada contém um inteiro  $N$ . Cada uma das próximas  $N$  linhas contém um dígito e um caractere ‘\*’ ou ‘/’, que representam uma operação realizada na calculadora.

## Saída

Seu programa deve imprimir uma única linha contendo o resultado que deve ser exibido pela calculadora ao final das operações.

## Restrições

- $1 \leq N \leq 100\,000$ .
- Os números informados são inteiros entre 1 e 9.
- O resultado **final** da conta é um número inteiro entre 1 e  $2^{30}$ .

## Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 20 pontos,  $N = 3$ .
- Em um conjunto de casos de teste que totaliza 50 pontos, o resultado da expressão até a operação  $i$  é um inteiro entre 1 e  $2^{30}$ , para  $i = 1, 2, \dots, N$ .

