Maia – Aquecimento

2^a Seletiva Interna – 2010/2

Sevidor BOCA:

`http://192.168.0.21:8001` (acesso interno)

`http://200.19.106.219:8001` (acesso externo)

Organizadores:

Alexandre Gonçalves Silva, Roberto Silvio Ubertino Rosso Jr., Claudio Cesar de Sá
{*alexandre,rosso,claudio*} *at joinville dot udesc dot br*

Lembretes:

- É permitido consultar livros, anotações ou qualquer outro material impresso durante a prova.
- A correção é automatizada, portanto, siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa. Deve-se considerar entradas e saídas padrão.
- Procure resolver o problema de maneira eficiente. Se o tempo superar o limite pré-definido, a solução não é aceita. As soluções são testadas com outras entradas além das apresentadas como exemplo dos problemas.
- Teste seu programa antes de submetê-lo. A cada problema detectado (erro de compilação, erro em tempo de execução, solução incorreta, formatação imprecisa, tempo excedido ...), há penalização de 20 minutos. O tempo é critério de desempate entre duas ou mais equipes com a mesma quantidade de problemas resolvidos.
- Utilize o *clarification* para dúvidas da prova. Os juízes podem opcionalmente atendê-lo com respostas acessíveis a todos.

Problema A: Aritmética primária

`Arquivo: vaium.[c|cpp|java]`

As crianças são ensinadas a somar números com vários dígitos a partir da direita para a esquerda, um dígito por vez. Muitos acham a operação "vai 1" (ou "carry") – sendo esta feita a partir da posição de um dígito adicionado ao próximo – um desafio significativo. Seu trabalho é contar o número de operações "vai um" para cada conjunto de problemas de soma para que os educadores possam avaliar esta dificuldade.

Cada linha da entrada contém dois números inteiros sem sinal com menos de 10 dígitos. A última linha de entrada contém 0 0. Para cada linha de entrada, exceto a última que deve calcular e imprimir o número de operações "vai um" que possam resultar da adição de dois números, no formato indicado abaixo.

Exemplo de entrada

```
123 456
555 555
123 594
0 0
```

Exemplo de saída

```
No carry operation.
3 carry operations.
1 carry operation.
```

Problema B: Sequência crescente

`Arquivo: cresce.[c|cpp|java]`

Dada uma sequência de dígitos, inserir vírgulas para criar uma sequência de números estritamente crescente, de modo a minimizar a magnitude do número anterior. Para este problema, os zeros à esquerda são permitidos na frente de um número.

Especificação da entrada

A entrada será composta por vários casos de teste. Cada caso consiste de uma linha, contendo uma sequência de dígitos de comprimento máximo de 80. A linha contendo um único 0 finaliza a entrada.

Especificação da saída

Para cada instância, deve-se produzir uma sequencia estritamente crescente separada por vírgula, sem espaços entre vírgulas ou números. Se houver várias sequências, escolha aquela que possui o maior primeiro valor; se houver um empate, o maior segundo número, etc.

Exemplo de entrada

```
3456
3546
3526
0001
100000101
0
```

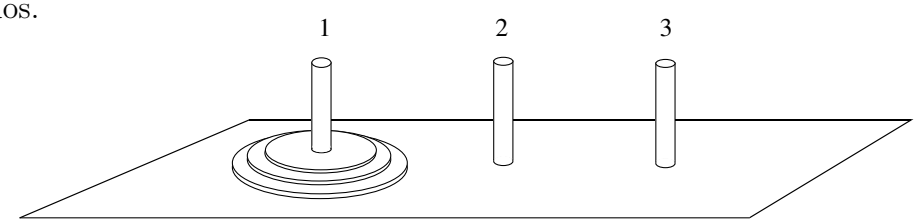
Exemplo de saída

```
3,4,5,6
35,46
3,5,26
0001
100,000101
```

Problema C: Torres de Hanói

`Arquivo: hanoi.[c|cpp|java]`

O quebra-cabeças Torres de Hanói é muito antigo e conhecido, sendo constituído de um conjunto de N discos de tamanhos diferentes e três pinos verticais, nos quais os discos podem ser encaixados.



Cada pino pode conter uma pilha com qualquer número de discos, desde que cada disco não seja colocado acima de outro disco de menor tamanho. A configuração inicial consiste de todos os discos no pino 1. O objetivo do quebra-cabeças é mover todos os discos para um dos outros pinos, sempre obedecendo à restrição de não colocar um disco sobre outro menor. Um algoritmo para resolver este problema é o seguinte.

```
1  procedimento Hanoi(N, Orig, Dest, Temp)
2      se N = 1 então
3          mover o menor disco do pino Orig para o pino Dest;
4      senão
5          Hanoi(N-1, Orig, Temp, Dest);
6          mover o N-ésimo menor disco do pino Orig para o pino Dest;
7          Hanoi(N-1, Temp, Dest, Orig);
8      fim-se
9  fim
```

Sua tarefa é escrever um programa que determine quantos movimentos de trocar um disco de um pino para outro serão executados pelo algoritmo acima para resolver o quebra-cabeça. A entrada possui vários conjuntos de teste. Cada conjunto de teste é composto por uma única linha, que contém um único número inteiro N ($0 \leq N \leq 30$), indicando o número de discos. O final da entrada é indicado por $N = 0$. Para cada conjunto de teste, o seu programa deve escrever três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato "Teste n", onde n é numerado sequencialmente a partir de 1. A segunda linha deve conter o número de movimentos que são executados pelo algoritmo dado para resolver o problema das Torres de Hanói com N discos. A terceira linha deve ser deixada em branco. A grafia mostrada no exemplo de saída, abaixo, deve ser seguida rigorosamente.

Exemplo de entrada

```
1
2
0
```

Exemplo de saída

```
Teste 1
1

Teste 2
3
```