# ToPAS'08
# Torneio de Programação para Alunos do Secundário

## Departamento de Ciência de Computadores

http://www.dcc.fc.up.pt/topas/

## Conjunto de Problemas


Faculdade de Ciências da Universidade do Porto

9 de Maio de 2008

Este conjunto de problemas deverá conter sete (7) problemas e quinze (15) páginas.
Se faltar algum problema, por favor avise a organização.

# ToPAS'08
# Torneio de Programação para Alunos do Secundário

Dep. Ciência de Computadores – FCUP
9 de Maio de 2008

## Contents

# Problema A

## Preciosismos

Dois amigos que já não se viam há alguns meses encontraram-se duas vezes no mesmo dia, às 7:58 e às 14:05. Um deles exclamou "Passaram apenas 367 minutos!" e o outro retorquiu "Queres dizer, 6 horas e 7 minutos?!". Mas, se se tivessem reencontrado às 7:59, o primeiro diria "Passou apenas 1 minuto!". O segundo diria "De facto!". Diria sempre isso se o intervalo fosse inferior a uma hora. Se se tivessem reencontrado às 13:58, o segundo diria "Queres dizer, 6 horas?!" e se fosse às 8:59 já seria "Queres dizer, 1 hora e 1 minuto?!". Pretendemos escrever diálogos idênticos a estes.

### Tarefa

Escrever um programa que produza as duas frases do diálogo para dois encontros (consecutivos), registados às $h_1$ horas e $m_1$ minutos e $h_2$ horas e $m_2$ minutos, sendo $h_1, m_1, h_2, m_2$ inteiros tais que $0 \leq h_i \leq 23$ e $0 \leq m_i \leq 59$, para $i = 1, 2$. Os valores de $h_i$ e $m_i$ são dados numa linha separados por um espaço. Cada uma das duas frases do diálogo é escrita numa linha diferente, terminando por mudança de linha.

### Exemplo 1

**Input**

```
7 58
14 5
```

**Output**

```
Passaram apenas 367 minutos!
Queres dizer, 6 horas e 7 minutos?!
```

### Exemplo 2

**Input**

```
2 1
2 48
```

**Output**

```
Passaram apenas 47 minutos!
De facto!
```

## Exemplo 3

**Input**

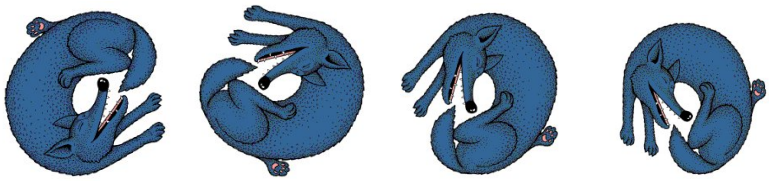
```
8 0
9 1
```

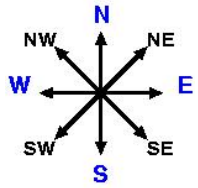
**Output**

```
Passaram apenas 61 minutos!
Queres dizer, 1 hora e 1 minuto?!
```

# Problema B

## Caçando Caudas






Por pura diversão ou algum tipo de perturbação, é habitual cães e gatos rodopiarem freneticamente, enquanto tentam apanhar a sua própria cauda (*"be chasing my tail"* quer dizer estar ocupado com mil e um afazeres, mas pouco sucesso). Com tantas voltas, é de imaginar que ao fim de algum tempo fiquem tontos, que ora rodem para um lado ora rodem para o outro, etc. Pretendemos saber qual a posição, o sentido de movimento e o estado em que se encontraria um animal depois de ter efectuado uma sequência de movimentos, considerando que:

- inicialmente, o animal está bem (estado `normal`) e tem a cabeça a Sul (`S`);
- cada movimento corresponde a uma rotação dum múltiplo de 45°; pode ser efectuado em sentido horário (`CW`) ou anti-horário (`CCW`); o primeiro movimento será no sentido anti-horário;
- se a amplitude dum dado movimento exceder 720° (duas voltas), o animal fica `tonto` e passa a rodar em sentido contrário; voltará ao estado `normal` logo que rodar novamente (se nesse movimento não rodar outra vez demais).

### Tarefa

Escrever um programa que, dada uma sequência de movimentos, indique a posição final da cabeça do animal (`N`, `S`, `E`, `W`, `NE`, `SE`, `NW`, ou `SW`), o sentido em que o animal iria rodar (`CW` ou `CCW`), e o seu estado final (`normal` ou `tonto`). Cada linha de *input* tem um inteiro positivo múltiplo de 45 (e inferior a 2000), o qual representa a amplitude da rotação, com excepção da última linha que tem `-1`. Haverá pelo menos um movimento. O programa escreve um terno `(posição, sentido, estado)`, com um espaço após cada vírgula, seguindo-se mudança de linha.

### Exemplo 1

**Input**

```
180
360
360
-1
```

**Output**

```
(N, CCW, normal)
```

## Exemplo 2

**Input**

```
540
405
765
45
810
855
810
810
-1
```

**Output**

```
(W, CW, tonto)
```

# Problema C

## Psittacus Erithacus

Reina grande confusão na quinta desde que Jacó chegou. Jacó aprendeu a imitar todos os animais na perfeição – "piupiu", "muuu", "miao", "cocorocoo", "cacaraca", "quaqua", "auau", "meemee", e "jaco". Um extenso repertório! e até faz composições com duas ou três palavras ("meemeeauau", "piupiumuu", "auaujacoquaqua", "cocorocoocacaraca", etc). O dono começou a achar melhor oferecer Jacó, mas a sua afeição ao bicho impedia-o de tomar uma decisão. Num dia de menos trabalho, fez um registo da situação. Sem se enganar, anotou o número de patas do animal que fez barulho e o barulho que fazia: *2piupiu2meemee4muuu4miao2cocorocoo4meemee2jacocacaraca2auau2jaco2miao 2piupiuquaquajaco4miao4auau2piupiumeemee4muuu4meemee2piupiu2cacaraca*... No fim, pensando melhor, viu que não sabia quais eram os registos do papagaio. Mas, nem tudo estava perdido.

### Tarefa

Escrever um programa que, dada uma sequência de registos terminada por **0**, cada um dos quais numa linha diferente, indique o número de registos total e o número mínimo e máximo dos do papagaio (escritos numa linha, separados por um espaço, e terminando com mudança de linha).

### Exemplo 1

**Input**

```
2 piupiu
2 meemee
4 muuu
2 jacoquaqua
4 auau
4 miao
0
```

**Output**

```
6 2 3
```

## Exemplo 2

### Input

```
2 piupiu
2 auau
2 jaco
2 piupiumiaoquaqua
4 meemee
4 miao
2 auaujacomiao
4 muuu
0
```

### Output

```
8 4 5
```

# Problema D

## Olh-Ó-Robô

Uma empresa tem um armazém robotizado: 2 robôs colocam e retiram as mercadorias das prateleiras sem qualquer intervenção humana. O armazém é rectangular e cada um dos robôs está atribuído a um dos lado. À partida os robôs nunca se encontrariam mas, por um erro de concepção, podem encontrar-se num único ponto. No *centro* do armazém está instalado o posto de carregamento das baterias que é comum a ambos os robôs. Os robôs aproveitam para ir a este sítio quando estão com as baterias em baixo e suficientemente perto do centro do armazém. Quando resolvem carregar baterias simultaneamente colidem e é necessária a intervenção humana.

Os robôs são comandados por instruções que partem do princípio que o armazém é uma grelha regular de 30 por 15 células, com o ponto de abastecimento nas coordenadas (0,0). Uma sequência de instruções é executada sem ciclos nem saltos, indicando apenas os movimentos do robô. Os comandos são apenas os números inteiros 2, 4, 5, 6 e 8, representando as direcções, como no teclado numérico: o 2 aponta para Sul, o 8 para Norte, o 4 para Oeste e o 6 para Este; o 5 significa permanecer no mesmo lugar (este comando pode ser executado em qualquer instante). A sequência de instruções é precedida pela localização do robô e terminada por 0. Os robôs estão sincronizados e a cada segundo executam um comando, começando exactamente ao mesmo tempo. Sabe-se que, durante a execução dos comandos, o robô vai *no máximo uma única vez* ao posto de carregamento das baterias e fica aí durante alguns segundos (executando o comando 5).

### Tarefa

Vai ser necessário escrever um programa que analise as sequências de comandos de ambos os robôs e verifique se podem ocorrer colisões; isto é, se os intervalos de tempo em que os robôs estariam a abastecer se sobrepõem. O programa recebe duas sequências de instruções que têm exactamente o mesmo comprimento e garantem já que os robôs se deslocam no interior do armazém. Cada uma delas é precedida da posição inicial do robô e terminada por 0. Se levarem à colisão dos robôs então o programa de verificação escreve `KO`. Se não existir perigo de colisão, escreve `OK`.

## Exemplo

### Input

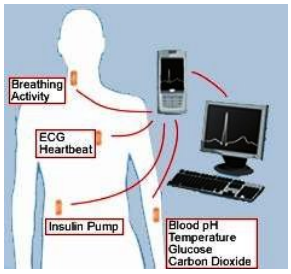
```
-7 0
6
6
6
6
6
6
6
5
5
0
7 0
4
4
4
4
4
4
4
5
5
0
```

### Output

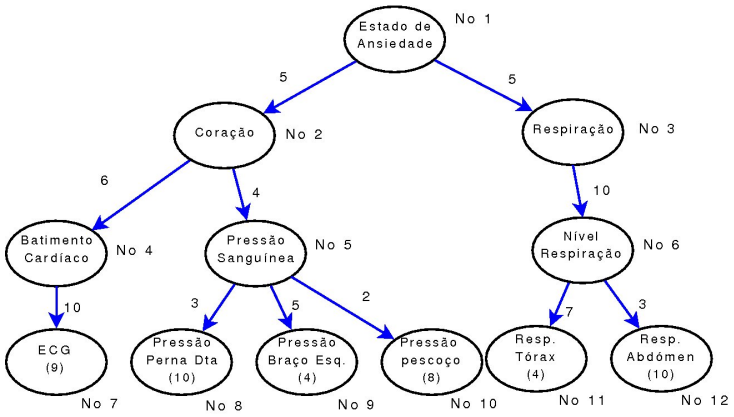
```
KO
```

# Problema E

## Ti-Nó-Ni




Num futuro não muito longínquo teremos sensores dentro e fora do nosso corpo que permitirão monitorizar e auxiliar a diagnosticar as doenças que nos afligem. Já não é de todo ficção científica a possibilidade de um doente ir para casa com um conjunto de sensores e um PDA para ser acompanhado à distância pelos seus médicos (que, eventualmente, lhe ligarão para o telemóvel a dizer que não está bem). Como se pode determinar o estado do doente a partir das leituras dos sensores? Usando uma árvore de decisão, por exemplo, como a da figura, para obter o indicador de doença (no exemplo, seria o valor do indicador de *Estado de Ansiedade*).




Processa-se a árvore de baixo para cima para obter um valor para cada nó. Esse valor vai ser a *parte inteira* da média pesada dos valores dos nós de que depende imediatamente (nós filhos). O peso de cada filho é definido pelo valor que está no ramo que o liga ao pai. O peso total dos filhos será sempre 10. Os nós que não têm filhos são os que correspondem aos sensores (e chamam-se folhas). Na figura, a leitura do sensor está indicada no nó entre parentesis. Assim, por exemplo, para calcular o *Nível Respiração*, de acordo com a figura, teriamos:

$$\textit{Nível Respiração} = (7 \times \textit{Resp. Tórax} + 3 \times \textit{Resp. Abdómen})\,/10 = (7 \times 4 + 3 \times 10)\,/10 = 5.8 \approx \mathbf{5}$$

Logo, também *Respiração* = 5 pois *Respiração* = $(10 \times$ *Nível Respiração*$)\,/10$. Se a árvore só tivesse um nó, esse nó correspondia a um sensor e a sua leitura dava o indicador da doença.

## Tarefa

Pretende-se implementar um programa que, dada a especificação duma árvore de decisão e os valores das leituras dos sensores, escreva o "*valor*" da doença do paciente, isto é o valor do nó 1 da árvore (seguido de mudança de linha).

Na primeira linha de *input* terá dois inteiros `n` e `f`, separados por espaço: `n` é o número total de nós e `f` o de folhas. Cada uma das `n-1` linhas seguintes define um ramo: `Pai Filho Peso`. Em cada uma das `f` linhas seguintes terá um par de valores `Sensor Leitura`. Os nós foram numerados de 1 a `n` de forma que cada filho tivesse um número superior ao seu pai e os sensores tivessem números de `n-f+1` a `n`. Os ramos estão ordenados por ordem crescente de número de nó pai. O número de nós é maior ou igual a 1 e não excede 100. Os valores (lidos ou calculados) também serão relativamente pequenos.

## Exemplo

### Input

```
12 6
1 2 5
1 3 5
2 4 6
2 5 4
3 6 10
4 7 10
5 8 3
5 9 5
5 10 2
6 11 7
6 12 3
7 9
8 10
10 8
11 4
12 10
9 4
```

### Output

```
6
```

# Problema F
## Barbearia Chic

(Foto de *Michel Waldmann*)

A Barbearia Chic não é grande mas é muito asseada. Três cadeiras, onde os clientes aguardam a sua vez antes de passarem ao cadeirão do barbeiro, e pouco mais lá cabe. O barbeiro não gosta nada de ter clientes em pé a estorvar. Por isso, quando não há lugares vagos, os clientes não entram! Mas, podem entrar se chegarem no momento em que outro fica pronto, desde que a Chic esteja aberta. A Chic abre às 9:00 e encerra às 12:00 para almoço. Reabre às 15:00 e encerra às 19:00. Os clientes que cheguem até às 12:00 ou até às 19:00, inclusivé, são atendidos, se a Chic não estiver cheia. O atendimento é feito por ordem de chegada. Os clientes não esperam se o barbeiro estiver livre e não ficam à espera à porta quando a Chic está fechada ou cheia. Não há mãos a medir para não perder clientes nem os fazer esperar muito. Pretende-se simular o funcionamento da Chic num certo dia, supondo que em cada instante chegará no máximo um cliente e que o barbeiro não dispende mais de quarenta minutos com cada cliente (o que lhe dá sempre algum tempo para almoçar).

### Tarefa

Escrever um programa para determinar o número de clientes que não puderam entrar, e que escreverá `Perdeu = r`, seguido de mudança de linha, sendo `r` esse número. O programa analisará uma sequência ordenada de `K` ocorrências `H M D`, em que `H` e `M` definem o instante em que o cliente chegou (`H` horas e `M` minutos) e `D` o tempo (em minutos) que o barbeiro dispenderia a atender o cliente se o puder fazer. Assim, `9 2 10` indica que o cliente chegou às 9:02 e o barbeiro gastaria 10 minutos, caso o atendesse. O valor de `K` é dado na primeira linha.

### Exemplo 1

**Input**

```
11
9 2 10
```

```
9 5 20
9 7 10
9 20 25
9 22 10
9 25 15
16 30 12
18 0 20
18 2 15
18 20 35
19 5 10
```

**Output**

```
Perdeu = 2
```

## Exemplo 2

**Input**

```
20
9 2 10
9 5 20
9 7 10
9 20 25
9 22 10
10 15 15
10 35 10
11 55 10
12 0 10
14 30 15
15 5 10
15 17 10
15 30 15
16 40 40
16 45 17
17 0 12
17 35 15
18 21 25
18 40 10
18 45 20
```

**Output**

```
Perdeu = 1
```

# Problema G

## Genial!

No jogo de tabuleiro *Genial!* cada um de quatro jogadores obtém pontos em seis cores: vermelho, verde, azul, laranja, amarelo e roxo. O conhecimento das regras deste jogo não é importante para o problema que se segue. A posição relativa de cada jogador no final do jogo é determinada pela pontuação na cor em que tiver o *menor* número de pontos. Ganha o jogador que tiver mais pontos na sua *pior* cor. Quando dois jogadores tiverem igual pontuação na pior cor, faz-se o desempate pela 2ª pior cor e assim sucessivamente. Por exemplo: sejam as pontuações dadas pela tabela:

| | Vermelho | Verde | Azul | Laranja | Amarelo | Roxo |
|---|---|---|---|---|---|---|
| Joana | 11 | 18 | 8 | 17 | 18 | 14 |
| Pedro | 8 | 18 | 18 | 9 | 18 | 18 |
| Miguel | 11 | 18 | 18 | 18 | 18 | 18 |
| Beatriz | 13 | 12 | 18 | 18 | 15 | 18 |

A Beatriz será vencedora (com 12 pontos) e em segundo lugar fica o Miguel (com 11 pontos). A Joana e o Pedro têm a mesma pontuação na pior cor (8 pontos), mas a Joana tem melhor resultado na 2ª menor (11 pontos contra 9 do Pedro); a Joana fica em terceiro e o Pedro em último lugar.

### Tarefa

Escrever um programa que leia as pontuações dos jogadores e escreva a ordem relativa entre estes. É dada a tabela de pontuações: 4 linhas; cada linha tem as 6 pontuações de um jogador (são inteiros positivos). O programa deve escrever uma única linha com a ordem relativa entre os jogadores, do melhor para o pior classificado. Não haverá empates. Identificamos os jogadores por números de 1 a 4: o jogador 1 tem as pontuações da primeira linha, o jogador 2 as da segunda, etc.

### Exemplo

#### Input

```
11 18 8 17 18 14
8 18 18 9 18 18
11 18 18 18 18 18
13 12 18 18 15 18
```

#### Output

```
4 3 1 2
```