Programação Imperativa
LEI (1º ano) – Ano Lectivo 2007/08

2ª fase de avaliação prática

7 de Maio de 2008

Apresentação e Normas

Esta é a segunda componente de avaliação contínua da disciplina de Programação Imperativa do 1º ano de LEI. Consiste numa prova prática com a duração de 1h30, na qual os grupos (formados por 3/2 alunos) têm de resolver 3 problemas que testam os princípios básicos da programação em C, nomeadamente a sintaxe, a implementação de funções e a utilização de *arrays* unidimensionais.

O programa em C que resolve cada problema deve ser submetido para avaliação automática, ao sistema Mooshak acessível no URL:

`http://mooshak.di.uminho.pt/~mooshak`

O sistema Mooshak classifica a submissão de **errada** (*compiler error*, *presentation error* ou *wrong answer*) ou de **correcta** (*accepted*). Um programa é considerado correcto se passar nos 6 testes a que é submetido. Para isso, o programa tem de produzir um resultado exactamente igual ao previsto, com base nas seguintes regras:

1. O programas deve ler do *standard-input* (sem qualquer pergunta inicial) e escrever no *standard-output*.

2. Os *inputs* usados para testar os programas submetidos obedecem às seguintes normas:

   - Não há espaços no fim de cada linha e todas as linhas terminam com mudança de linha (*newline*).
   - Não se usa espaçamento múltiplo, a menos que o enunciado do problema respectivo o especifique explicitamente.

3. Os *outputs* devem obedecer às mesmas normas do ponto anterior.

O enunciado dos problemas a resolver contem implementações da operação de leitura de dados necessárias à sua resolução.

Um problema errado pode ser re-submetido para nova avaliação.

Para evitar qualquer surpresa na compilação automática do programa, aquando da sua submissão ao sistema de avaliação, informa-se que o Mooshak compila os programas C usando o seguinte comando (em que `$file` é o nome do ficheiro contendo o programa submetido):

```
/usr/bin/gcc -Wall -lm -O2 $file
```

Por cada submissão, o sistema Mooshak regista apenas o **login do grupo** (regista portanto o número do grupo e não o nome/número de cada um dos seus elementos) e o **endereço IP** do computador do qual foi submetida a resolução do problema.

Problema A

Os problemas que se apresentam pretendem processar o *log-file* de uma prova de velocidade.

Cada um destes ficheiros (*log-file*) consiste em várias linhas (de texto) em que cada uma tem a forma

```
evento concorrente tempo
```

em que

- `evento` é um caracter (`'P'` ou `'C'`) que indica se se trata de uma `P`artida ou de uma `C`hegada.
- `concorrente` é um número inteiro que identifica o concorrente
- `tempo` é o tempo em mili-segundos em que o evento ocorreu (número inteiro não negativo)

Por exemplo, um exemplo de um destes *log-files* seria

```
P 1 0
P 2 3110
P 3 4211
C 2 5315
P 4 6418
C 1 7520
C 4 8625
```

Para todos os problemas que a seguir se apresentam, assuma o seguinte:

- O log-file, respeitando o formato acima descrito, é passado ao seu programa pelo *standard input*.
- Pode haver concorrentes que nunca chegam a acabar a prova i.e., têm um tempo de partida mas não têm tempo de chegada.
- Pode haver concorrentes para os quais exista um tempo de chegada e não exista um tempo de partida, considerando-se neste caso que o carro não efectuou uma prova válida.
- O tempo de chegada é sempre superior ao tempo de partida.

Caso opte por usar *arrays* em qualquer um dos seguintes problemas, terá uma penalização de 30% sobre a cotação da pergunta. A não utilização de *arrays* será verificada automaticamente pelo Mooshak através da existência, ou não, de parênteses rectos (‘[’ ou ‘]’) no código fonte dos programas submetidos.

Problema X

Desenvolva um programa que devolva o número de concorrentes que acabaram a prova, ou seja, que tenham um tempo de partida e um tempo de chegada.

Problema Y

Implemente um programa que calcule a média dos tempos de percurso de todos os concorrentes que acabaram a prova. Este programa tem de devolver como resultado um valor do tipo `int`. Caso seja necessário efectue o *cast* para o tipo `int`.

Problema Z

Desenvolva um programa que devolva os três primeiros classificados na prova. O programa deve considerar apenas os concorrentes que terminaram a prova, podendo assim dar-se o caso de haver apenas 2, 1 ou até nenhum vencedor. Caso hajam concorrentes com tempos iguais, ganha o que tiver partido primeiro.

Problema B

Desenvolva um programa que efectue a multiplicação de duas matrizes compatíveis de valores inteiros.

A representação de matrizes para leitura segue o seguinte formato:

```
Linhas Colunas
Valores da matriz
```

Este formato pode ser lido pela seguinte função.

```
typedef int Matriz[100][100];

void readMatriz(Matriz m, int* linhas, int* colunas) {
  int i, j;

  scanf("%d %d\n", linhas, colunas);
  for(i=0; i < *linhas; i++)
    for(j=0; j < *colunas; j++)
      scanf("%d", &m[i][j]);
}
```

O formato de escrita das matrizes é efectuado com a sintaxe do Haskel para valores do tipo `[[Int]]`, os quais podem ser obtidos através da seguinte função.

```
void showMatriz(Matriz m, int linhas, int colunas) {
  int i,j;

  printf("[");
  for(i=0; i < linhas; i++) {
    printf("[");
    for(j=0; j < colunas; j++) {
      printf("%d", m[i][j]);
      if(j != colunas -1) printf(",");
    }
    printf("]");
    if(i != linhas -1) printf(",");
  }
  printf("]\n");
}
```

A titulo de exemplo de execução do programa pedido, quando este recebe o seguinte input

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

terá de dar como resultado o seguinte output

```
[[14,32,32],[32,77,122]]
```