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

3ª fase de avaliação prática

11 de Junho de 2008

Apresentação e Normas

Esta é a terceira componente de avaliação prática da disciplina de Programação Imperativa do 1° ano de LEI.

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 1

Os problemas que se apresentam tem como objectivo processar o *log-file* de uma prova de velocidade e devolver o resultado de diversos cálculos sobre o seu conteúdo. Ao contrário do problema semelhante a este apresentado na segunda prova de Mooshak, agora, o ficheiro de registo de tempos de cada carro pode contemplar também os tempos de três etapas intermédias (Etapa X, Etapa Y e Etapa Z) entre as etapas de partida (P) e chegada (C).
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 `'X'` ou `'Y'` ou `'Z'` ou `'C'`) que indica se se trata de uma `P`artida, do tempo de passagem numa etapa (`X`, `Y`, ou `Z`) 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)

Um exemplo de um destes *log-files* em que o carro 1 e 2 efectuam um prova válida e o carro 3 efectua uma prova inválida por não ter passado nas etapas X e Y, seria

```
P 1 0
C 3 7432
P 3 1231
X 1 1231
C 2 5315
Z 3 3214
C 1 7520
Y 1 2138
X 2 2111
Z 1 5678
P 2 1234
Z 2 4213
Y 2 3215
```

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 não tenham um tempo para uma qualquer etapa (P, X, Y, Z ou C). Nestes casos, a prova do carro em causa é considerada inválida.
- O tempo de um concorrente para uma etapa é sempre inferior ao tempo desse mesmo concorrente para uma outra etapa posterior.
- Não existe qualquer ordem nos registos apresentados no *log-file*.
- Há sempre pelo menos um carro que efectua uma prova válida.

Para efectuar a leitura do *log-file* pode utilizar a seguinte função:

```
void ReadCars() {
   char c;
   int carid, time;
   while(scanf("%c %d %d\n", &c, &carid, &time) == 3) {
       ...
   }
}
```

Pode ainda testar as suas soluções utilizando os seguintes ficheiros de vectores de teste como input.

Teste	**N.º Carros**	**Ficheiro input**
1	5	http://alfa.di.uminho.pt/∼nfr/Mooshak3Tests/P1-test1public.in
2	20	http://alfa.di.uminho.pt/∼nfr/Mooshak3Tests/P1-test2public.in
3	100	http://alfa.di.uminho.pt/∼nfr/Mooshak3Tests/P1-test3public.in

E os seguintes ficheiros de output para cada um dos ficheiros de teste e respectivos problemas apresentados.

Problema	**Teste**	**Ficheiro output**
1.A	1	http://alfa.di.uminho.pt/∼nfr/Mooshak3Tests/P1a-test1public.out
1.A	2	http://alfa.di.uminho.pt/∼nfr/Mooshak3Tests/P1a-test2public.out
1.A	3	http://alfa.di.uminho.pt/∼nfr/Mooshak3Tests/P1a-test3public.out
1.B	1	http://alfa.di.uminho.pt/∼nfr/Mooshak3Tests/P1b-test1public.out
1.B	2	http://alfa.di.uminho.pt/∼nfr/Mooshak3Tests/P1b-test2public.out
1.B	3	http://alfa.di.uminho.pt/∼nfr/Mooshak3Tests/P1b-test3public.out
1.C	1	http://alfa.di.uminho.pt/∼nfr/Mooshak3Tests/P1c-test1public.out
1.C	2	http://alfa.di.uminho.pt/∼nfr/Mooshak3Tests/P1c-test2public.out
1.C	3	http://alfa.di.uminho.pt/∼nfr/Mooshak3Tests/P1c-test3public.out

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 parêntesis rectos ('[' ou ']') no código fonte dos programas submetidos.

Problema A

Implemente um programa que calcule as médias dos tempos por etapa e devolve-os por ordem de ocorrência das etapas, ou seja, pela seguinte ordem $P \rightarrow X$, $X \rightarrow Y$, $Y \rightarrow Z$ e $Z \rightarrow C$. Para o cálculo destas médias, apenas contam os concorrentes com provas válidas. Este programa tem de devolver como resultado 4 valores do tipo `int`, um por cada linha no *output*. Caso seja necessário efectue o *cast* para o tipo `int`.

Problema B

Desenvolva um programa que devolva o tempo da etapa mais lenta do carro mais rápido a efectuar uma prova válida.

Problema C

Desenvolva um programa que devolva o menor tempo em que é possível efectuar a prova na totalidade, ou seja, que devolva a soma dos tempos mínimos por etapa independentemente de terem sido efectuados por carros com provas válidas ou não.

Problema Z

Defina uma função que efectua a leitura de um texto pelo *stdin*, e escreve no écran (*stdout*) esse mesmo texto com um único espaço após um ponto final ou virgula e a primeira letra de cada frase transformada na correspondente maiúscula. Note que o início de uma frase se detecta como o primeiro caracter não branco após um ponto final e que o último ponto final do texto nao deve ser seguido de qualquer espaço.
A titulo de exemplo, ao receber o seguinte texto como input

```
A Licenciatura em Engenharia Informática (LEI) tem por objectivo preparar
engenheiros informáticos,aptos em todas as fases actualmente reconhecidas no
rigoroso processo de análise,      concepção e implementação de soluções
informáticas.os seus licenciados adquirem durante o curso grande experiência
teórica e prática na análise de sistemas a informatizar,na especificação dos
requisitos desses mesmos sistemas e nas técnicas de construção de protótipos.
```

a sua função deverá retornar o seguinte texto como output

```
A Licenciatura em Engenharia Informática (LEI) tem por objectivo preparar
engenheiros informáticos, aptos em todas as fases actualmente reconhecidas no
rigoroso processo de análise, concepção e implementação de soluções
informáticas. Os seus licenciados adquirem durante o curso grande experiência
teórica e prática na análise de sistemas a informatizar, na especificação dos
requisitos desses mesmos sistemas e nas técnicas de construção de protótipos.
```

Nota: Pode testar se já não existe mais nada para ler no *stdin* através da instrução `feof(stdin)` ou testando se o valor retornado por `getchar()` é igual a `EOF`.