FEUP	FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO Mestrado em Engenharia Informática e Computação Fundamentos da Programação	Auto Teste AT09 FACULTATIVO Duração: 90 min.

NOTAS IMPORTANTES:

1 - Deve respeitar rigorosamente os nomes dos procedimentos que são indicados bem como os formatos de saída dos resultados.

2 - Não utilize carateres acentuados nos nomes dos procedimentos nem nos parâmetros.

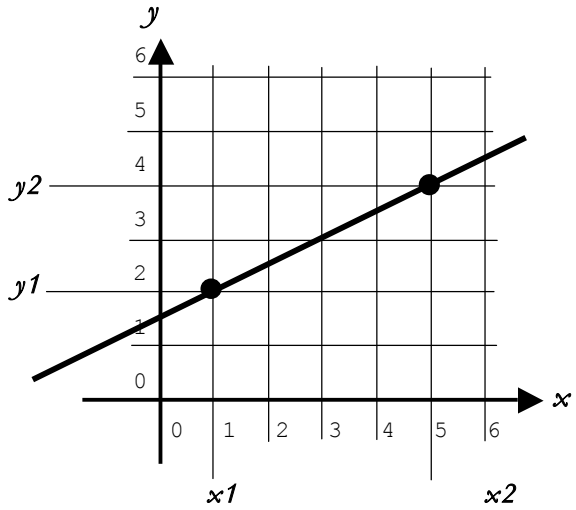
3 - Utilize comentários só "with Semicolons" ("com ponto e vírgula") e nunca "with a Box" ("com uma caixa").

4 - O código desenvolvido durante a prova, contido num único ficheiro com a extensão ".scm", deve ser submetido no Moodle usando o "link" correspondente à prova realizada. A não observação desta regra levará a que o código submetido não possa ser avaliado.

5 - Antes de submeter o ficheiro, assegure que este não tem erros de sintaxe (não dá erro ao premir o botão "correr").

6 - Assegure que o ficheiro não produz qualquer output (não há resultado visível no ecrã ao premir o botão "correr").

1- A função de uma reta é dada por **y = b + mx.**
1.1- Desenvolva o procedimento **funcao-reta** que recebe como argumentos dois pontos que definem a reta, representados pelos pares (x1 . y1) e (x2 . y2) e devolve um procedimento que implementa a função da reta correspondente.



$$y = b + m \cdot x$$

m = (y2 - y1) / (x2 - x1)
(inclinação da reta)

b = y1 - m . x1
(interseção da reta com o eixo dos

```
> (define r1 (funcao-reta (cons 1 2) (cons 5 4)))
> (r1 0.0)
1.5
> (r1 1.0)
2.0
> (r1 6.0)
4.5
> (r1 2.0)
2.5
> (define r2 (funcao-reta (cons 5 2) (cons 5 4)))
reta vertical
>
```

Atenção!
Caso em que os pontos estão na mesma vertical- termina com a visualização de APENAS "reta vertical"

1.2- Desenvolva o procedimento **cima?** que recebe como parâmetros **reta1** e **reta2**, os nomes das funções de duas retas, e a coordenada **x**, e determina se, para aquele **x**, **reta1** está acima da **reta2**, devolvendo #t ou #f, conforme o caso.

```
> (define r1 (funcao-reta (cons 1 2) (cons 5 4)))
> (define r2 (funcao-reta (cons 1 1) (cons 4 4)))
> (cima? r1 r2 2)
#t
> (cima? r1 r2 3)
#f
> (cima? r1 r2 2.5)
#t
>
```

(Continua →)

2- Uma liga de futebol pretende processar informação relativa aos resultados dos jogos realizados num campeonato de futebol e à pontuação dos clubes recorrendo a um programa a desenvolver na linguagem *Scheme*. O que se pretende neste problema é que desenvolva alguns dos procedimentos necessários nesse programa. Para simplificar, nos exemplos apresentados usar-se-ão como nomes dos clubes os símbolos 'a, 'b, 'c, … .

2.1- O procedimento **cria-jogo** tem como parâmetros os nomes dos clubes que disputaram um determinado jogo e os golos marcados por cada um, devolvendo um **par** (de *Scheme*) constituído por 2 **pares**, o 1º constituído pelo nome dos dois clubes e o 2º constituído pelos golos marcados por cada um deles.

```
> (define j1 (cria-jogo 'a 3 'b 2))
> j1
((a . b) 3 . 2)
```

Complete o procedimento **cria-jogo** :

```
(define cria-jogo
  (lambda (clube1 golos1 clube2 golos2) ...)
```

2.2- O procedimento **vencedor-jogo** tem como parâmetro um **jogo** e devolve o nome (**símbolo**) do clube vencedor ou, caso tenha ocorrido um empate, o símbolo **'empate**.

```
> (vencedor-jogo j1)   ; j1 que foi definido em 3.1
a
> (define j2 (cria-jogo 'c 1 'd 1))
> (vencedor-jogo j2)
empate
```

Complete o procedimento **vencedor-jogo** :

```
(define vencedor-jogo
  (lambda (jogo)  ...)
```

2.3- O procedimento **cria-pontuacao** tem como parâmetros uma **lista** com os nomes dos clubes e devolve uma outra **lista** cujos elementos são **pares**, sendo cada **par** constituído pelo nome do clube e pelo valor zero (a pontuação inicial de todos os clubes).

```
> (cria-pontuacao '(a b c d))   ; note que o nº de clubes podia ser outro
((a . 0) (b . 0) (c . 0) (d . 0))
```

Complete o procedimento **cria-pontuacao** :

```
(define cria-pontuacao
  (lambda (clubes) ...)
```

2.4- O procedimento **atualiza-pontuacao** tem como parâmetros uma **lista** do tipo da que é devolvida pelo procedimento **cria-pontuacao** (ver 2.3), o nome (**símbolo**) de um clube e os pontos conquistados por esse clube numa determinada jornada, e devolve uma **lista** contendo a pontuacao de todos os clubes atualizada para o clube passado como argumento (nota: a ordenação inicial da lista deve ser mantida). Admita que os argumentos passados a este procedimento são sempre válidos.

```
> pontuacao-atual
((a . 13) (b . 4) (c . 7) (d . 10))
> (atualiza-pontuacao pontuacao-atual 'c 3)
 ((a . 13) (b . 4) (c . 10) (d . 10))
```

Complete o procedimento **atualiza-pontuacao** :

```
(define atualiza-pontuacao
  (lambda (pontuacao clube pontos) ...)
```

2.5- O procedimento **primeiros** tem como parâmetro uma **lista** do tipo da que é devolvida pelo procedimento **cria-pontuacao** (ver 2.3) e devolve um **par** constituído pelo(s) clube(s) que têm a maior pontuação bem como o valor dessa pontuação, no formato ilustrado no exemplo seguinte. Caso a lista fornecida como argumento esteja vazia deve ser devolvido o símbolo **'pontuacao_invalida**.

```
> pontuacao-atual
((a . 13) (b . 4) (c . 13) (d . 10))
> (primeiros pontuacao-atual)
((a c) . 13)
> (primeiros '())
pontuacao_invalida  ; note que os carateres não são acentuados
```

Complete o procedimento **primeiros?** :

```
(define primeiros
  (lambda (pontuacao) ...)
```

---------- FIM da Prova Prática ----------