Universidade do Porto Faculdade de Engenharia FEUP	**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO** **Mestrado em Engenharia Informática e Computação** **Fundamentos da Programação - 2013/2014**	**Prova Prática 1** **Duração: 60 min.** **11 Nov 2013**

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-se de 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").

7 - Durante a prova apenas poderá aceder às páginas do Moodle de Fundamentos da Programação, com exceção dos fóruns. A prova prática será monitorizada e qualquer acesso a outro recurso do Moodle que não os de Fundamentos da Programação deste ano letivo implicará a anulação da prova.

Caminhos de Portugal

Acabou de ser contratado como o novo programador para o processo de informatização da Caminhos de Portugal. As suas primeiras tarefas consistem em desenvolver um conjunto de procedimentos para o sistema informático da plataforma. Os clientes da plataforma ganham pontos de fidelização que depois podem converter em descontos.

1 – (10 valores em 100)

Pretende-se desenvolver o procedimento **pontos->euros** que calcule e devolva quantos euros vale um determinado montante em pontos. Para tal basta usar a seguinte fórmula:

$$euros = \frac{pontos * 2.8}{10}$$

Observe os seguintes exemplos de utilização deste procedimento:

```
> (pontos->euros 10)
2.8
> (pontos->euros 25)
7.0
> (pontos->euros 110)
30.8
```

Completar o procedimento **pontos->euros**:

```
(define pontos->euros
 (lambda (pontos)
       ...
```

2 - (20 valores em 100)

Desenvolva o procedimento **converte-pontos** que devolve a quantia em euros necessária para comprar um determinado bilhete. O procedimento deve receber um montante de pontos disponíveis (**pontosdisp**) e o valor do bilhete em euros (**precobilhete**). Se o valor em euros, correspondente aos pontos disponíveis, cobrir o preço do bilhete o procedimento deve devolver 0, caso contrário, deve devolver o montante em euros remanescente a pagar.

Obs.: Pode utilizar o procedimento **pontos->euros** *que acabou de definir na alínea anterior.*

Observe os seguintes exemplos de utilização deste procedimento:

```
> (converte-pontos 105 28)
0
> (converte-pontos 105 30)
0.6
> (converte-pontos 105 92.5)
63.1
```

Completar o procedimento **converte-pontos**:

```
(define converte-pontos
 (lambda (pontosdisp precobilhete)
       ...
```

(v.s.f.f.)

3 - (30 valores em 100)
Considere agora que pretende implementar um sistema de descontos baseado em idades. Crie um procedimento **classe-desconto** que devolva um número correspondente ao valor do desconto para uma determinada idade (**idade**) de um cliente. Para tal deverá utilizar a seguinte tabela de referência:

Idade	Desconto (%)
idade <= 7	50
7 < idade <= 18	40
18 < idade <= 23	25
23 < idade <= 64	0
idade >= 65	35

Observe os seguintes exemplos de utilização deste procedimento:
```
> (classe-desconto 3)
50
> (classe-desconto 42)
0
> (classe-desconto 95)
35
```

Completar o procedimento **classe-desconto**:
```
(define classe-desconto
 (lambda (idade)
       ...
```

Trabalhar com recursividade...

4 - (25 valores em 100)
Pretende-se agora implementar o sistema de fidelização. Cada vez que um passageiro viaja com a companhia recebe um bónus de 10% aos pontos que já possui.

Desenvolva o procedimento **calcula-pontos-bonificados** que, com base nos **pontos** do cliente e do número de viagens (**numviagens**) que o mesmo tem registadas no sistema de base de dados da companhia, calcule e devolva o número de pontos final.

Obs.: Repare que não basta somar a percentagem do desconto pois na segunda viagem os 10% são sobre o total de pontos obtidos com a viagem anterior.

Observe bem os seguintes exemplos de utilização deste procedimento:
```
> (calcula-pontos-bonificados 100 0)
100
> (calcula-pontos-bonificados 100 1)
110.0
> (calcula-pontos-bonificados 100 2)
121.0
> (calcula-pontos-bonificados 100 3)
133.1
```

Completar o procedimento **calcula-pontos-bonificados**:
```
(define calcula-pontos-bonificados
 (lambda (pontos numviagens)
       ...
```

5- (15 valores em 100)
Pretende-se agora guardar um identificador único público para cada utilizador. No entanto, por razões de segurança e privacidade, este identificador não pode ser o número do CC ou de identificação fiscal do mesmo. Desta forma, decidiu-se recorrer a uma função criptográfica que funciona da seguinte forma:
Os números são codificados em conjuntos de dois dígitos formados pelo primeiro e último dígitos. Estes conjuntos são depois multiplicados e descartados do número original (o CC). O processo repete-se até que só haja um ou nenhum dígito. As parcelas são então somadas para dar o identificador final.

Exemplo:
12345678 é convertido em: 1*8 + 2*7 + 3*6 + 4*5
123 é convertido em: 1*3 + 2*0 (para número de dígitos ímpar, ignore o dígito do meio)

Ou seja, o procedimento deve receber um número (**12345678)** no exemplo acima e começar por obter o número mais à esquerda e mais à direita (**1** e **8** neste exemplo) e multiplicá-los. A função deve depois repetir este processo para o número sem os dígitos já multiplicados (i.e. para o número **234567**).

Observe os seguintes exemplos de utilização deste procedimento:
```
> (identificador-unico 12345678)
60
> (identificador-unico 123)
3
> (identificador-unico 1)
0
```

Completar o procedimento **identificador-unico**:
```
(define identificador-unico
 (lambda (n)
       ...
```