FEUP	FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO Mestrado em Engenharia Informática e Computação Fundamentos da Programação	Auto Teste AT03 FACULTATIVO Duração: 60 min.

NOTAS IMPORTANTES:

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

2 - Não use nunca carateres acentuados, nem nos nomes dos procedimentos nem dos parâmetros.

3 - Utilize comentários só "with Semicolons" e nunca "with a Box".

4 - O código desenvolvido durante a realização da 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.

1- fruta e muita-fruta

1.1-

O procedimento **fruta** tem como parâmetros: **laranjas** e **peras**.
Os parâmetros representam o peso de laranjas e peras e são ambos positivos. Também podem assumir o valor zero. O procedimento fruta devolve:

1 - quando o peso de **laranjas** for **maior** que o peso de **peras**
2 - quando o peso de **laranjas** for **menor** que o peso de **peras**
3 - quando os pesos de **laranjas** e de **peras** forem **iguais, exceto quando iguais a zero**
0 - quando os pesos de **laranjas** e de **peras** forem **iguais a zero**

```
(define fruta
  (lambda (laranjas peras)
    ...
```

Complete o procedimento **fruta**.

1.2-

Agora, pretende-se detetar as situações em que algum dos pesos, por engano, surja negativo.
O procedimento **fruta-com-negativos** devolve:

- o mesmo que o procedimento fruta, **exceto** se algum dos parâmetros aparecer negativo. Nesta situação, devolve -1.

Nota: este procedimento **pode** usar o procedimento **fruta**. Neste sentido, pode considerá-lo disponível, mesmo que o não tenha desenvolvido.

```
(define fruta-com-negativos
  (lambda (laranjas peras)
    ...
```

Complete o procedimento **fruta-com-negativos**.

1.3-

O procedimento **muita-fruta** tem como parâmetros: **laranjas**, **peras**, **mangas**, **bananas**, e **figos**, que representam os pesos de cada tipo de fruta.
Os pesos não deveriam exibir valores negativos, mas podem ser zero.
O procedimento **muita-fruta** devolve:

1 - se a soma dos pesos de **laranjas** e **peras** for **maior** que a soma dos peso dos outros frutos
2 - se a soma dos pesos de **laranjas** e **peras** for **menor que ou igual** à soma dos peso dos outros frutos

No entanto, a devolução 1 ou 2 só ocorrerá se o **peso de** figos **for menor que o peso de** bananas... Se esta condição não se verificar, em vez de 1 devolve 3 e em vez de 2 devolve 4.
O procedimento devolve -1, se algum dos pesos for negativo.

```
(define muita-fruta
  (lambda (laranjas peras mangas bananas figos)
    ...
```

Complete o procedimento **muita-fruta**.

2- falsa numeração

No contexto de um confronto militar, para confundir o inimigo, alguém inventou uma "nova numeração", uma numeração anormal, baseada na seguinte sequência de dígitos decimais:

```
0, 2, 4, 6, 8, 1, 3, 5, 7, 9          <--------- numeração falsa
```

que, na numeração normal equivalem a:

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9          <--------- numeração verdadeira
```

Pretende-se processar e visualizar valores nesta falsa numeração.
Por exemplo, pretende-se um procedimento que some dois valores inteiros representados na falsa numeração e que devolva o resultado nessa mesma numeração.

(v.s.f.f.)

```
> (soma-falso 18 34)
223
> (soma-falso 18 83)
200
```

A estratégia seguida foi: converter os dois números para a numeração verdadeira e processá-los normalmente. Converter o resultado para a numeração falsa e, finalmente, devolvê-lo.

Desde já se identifica o procedimento **falso->verdadeiro**, que converte um número na representação falsa para a representação verdadeira. No final, devolve-o em numeração verdadeira.

2.1-

```
(define falso->verdadeiro
  (lambda (num-falso)
    ...
```

Complete o procedimento **falso->verdadeiro**.

Nota: este procedimento deve usar o procedimento **digito-f-verd** que é apresentado na alínea seguinte. Neste sentido, pode considerá-lo disponível, mesmo que o não tenha desenvolvido.

2.2-

Naturalmente, deve concluir que para o procedimento **falso->verdadeiro** será importante ter o procedimento **digito-f-verd**, que toma um dígito falso e converte-o em verdadeiro. No final, devolve-o

```
(define digito-f-verd
  (lambda (d-f)
    (cond ((= d-f 2) 1)
    ...
```

Complete o procedimento **digito-f-verd**.

DAQUI ATÉ AO FIM, A PROVA JÁ NÃO SERÁ AVALIADA!

2.3-

E também no sentido inverso... O procedimento **verdadeiro->falso** converte um número na representação verdadeira para a representação falsa. No final, devolve-o.

```
(define verdadeiro->falso
  (lambda (num-verd)
    ...
```

Complete o procedimento **verdadeiro->falso**.

Nota: este procedimento deve usar o procedimento digito-v-falso que é apresentado na alínea seguinte.
Neste sentido, pode considerá-lo disponível, mesmo que o não tenha desenvolvido.

2.4-

O procedimento **digito-v-falso** toma um dígito verdadeiro, converte-o em falso e devolve-o.

```
(define digito-v-falso
  (lambda (d-v)
    (cond ((= d-v 1) 2)

    ...
```

Complete o procedimento **digito-v-falso**.

2.5-

Finalmente, o procedimento **soma-falso** toma dois valores na representação falsa, converte-os na representação verdadeira e soma-os. Depois converte a soma obtida em representação falsa e devolve-a.

```
(define soma-falso
  (lambda (n-falso1 n-falso2)
    ...
```

Complete o procedimento **soma-falso**.

Nota: este procedimento deve usar os procedimentos que foram considerados nas alíneas de 2.1 a 2.4. Neste sentido, pode considerá-los disponíveis, mesmo que os não tenha desenvolvido.

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