Introdução à Programação I, 11 de Janeiro de 2002

Duração máxima 2 horas e 30 minutos; Com consulta

Nome (legível): _____ Número _____

Problema 1 (7.5 valores)

Uma ideia para tirar um elemento à sorte de uma lista, consiste no seguinte:

- determinar, de uma forma aleatória, a ordem do elemento a retirar de uma lista. Supor que ao primeiro elemento da lista corresponde a ordem 1, ao segundo a ordem 2, etc.;
- criar uma nova lista, equivalente à lista dada, depois de lhe retirar o elemento identificado à sorte.

1.1 Antes de mais, começar por completar o procedimento ***tira-um-de-1-a-n***.

Exemplos:

```
> (tira-um-de-1-a-n 1 (list 1 2 3 4))
(2 3 4)
> (tira-um-de-1-a-n 2 (list 1 2 3 4))
(1 3 4)
```

```
; devolve uma lista composta por todos os elementos de lis, à qual se retira o elemento
; de ordem n, sendo o primeiro elemento de ordem 1. O parâmetro n é um inteiro nunca
; maior que o comprimento da lista lis.
(define tira-um-de-1-a-n
  (lambda (n lis)
```

1.2 Agora, completar o procedimento ***tira-um-a-sorte***, sabendo que se baseia na ideia acima exposta e que utiliza o procedimento ***tira-um-de-1-a-n***.

Exemplos:

```
> (tira-um-a-sorte (list 1 2 3 4))
(1 2 4)                          (neste caso, foi tirado o terceiro elemento)
> (tira-um-a-sorte (list 1 2 3 4))
(1 2 3)                          (neste, foi tirado o quarto)
> (tira-um-a-sorte (list 1 2 3 4))
(1 2 4)                          (neste, foi tirado novamente o terceiro)
> (tira-um-a-sorte (list 1 2 3 4))
(2 3 4)                          (neste, foi tirado o primeiro)
```

```
(define tira-um-a-sorte
  (lambda (lis)
```

Problema 2 (7.5 valores)

Quando se joga no totoloto com apostas simples, só se podem escolher 6 números, de 1 a 49, por cada aposta. Se, por engano, forem escolhidos mais de 6 números, apenas serão considerados os 6 mais pequenos, pois os restantes são desprezados. Vamos imaginar que uma aposta é criada através do construtor ***cria-aposta-vazia***.

```
(define cria-aposta-vazia
  (lambda ()
    (list 0 0 0 0 0 0)))
```

Agora, analisar cuidadosamente o diálogo que se segue:

```
> (define ap1 (cria-aposta-vazia))        (criação de uma posta vazia,
> ap1                                                    com a designação ap1)
(0 0 0 0 0 0)
> (mais-um-numero! ap1 6)                 (escolha do primeiro número)
ok
> ap1
(6 0 0 0 0 0)
> (mais-um-numero! ap1 6)                 (tentativa de escolha do mesmo número...)
ok
> ap1
(6 0 0 0 0 0)
> (mais-um-numero! ap1 16)                (escolha do segundo número)
ok
> (mais-um-numero! ap1 17)                (escolha do terceiro número)
ok
> (mais-um-numero! ap1 33)                (escolha do quarto número)
ok
> ap1
(6 16 17 33 0 0)
> (mais-um-numero! ap1 13)                (escolha do quinto número)
ok
> (mais-um-numero! ap1 18)                (escolha do sexto número)
ok
> ap1
(6 16 17 33 13 18)
> (mais-um-numero! ap1 1)                 (tentativa de escolha de mais
ok                                                          um número...)
> ap1                                     (resultou, pois é menor que o
(6 16 17 1 13 18)                                    maior número até à data)
```

Como se pode verificar na parte final do diálogo, a tentativa de escolher mais um número quando a aposta já está completa com 6 números, apenas resulta se o número pretendido for inferior ao maior número dos 6 que constituem a aposta. Nessa situação, o número pretendido substitui o maior número dos 6.
Julgou-se que seria útil o modificador ***substitui!***, com os parâmetros ***lis*** (uma lista), ***velho*** (um valor qualquer), ***novo*** (um valor qualquer). O modificador procura ***velho*** na lista e, se o encontrar, substitui-o por ***novo***. Se ***velho*** não existir na lista, então fica tudo na mesma.

2.1 Completar o modificador ***substitui!***.

```
(define substitui!
  (lambda (lis velho novo)
```

2.2 Completar o modificador ***mais-um-numero!***, sabendo que utiliza ***substitui!***. Ainda se sabe que o modificador que se pretende completar não controla se ***num*** é inteiro e se se situa entre 1 e 49.

```
(define mais-um-numero!
  (lambda (aposta num)
    (cond ((member num aposta) 'ok)
```

Problema 3 **(5.0 valores)**

Há uns anos atrás encontrava-se com alguma frequência o *Jogo dos Furos*, associado à venda, por exemplo, de chocolates. A base do jogo era uma folha de cartolina, com pequenos furos organizados em matriz, escondidos com uma folha de papel colada à cartolina; apesar de escondidos, a posição dos furos era bem visível, o que já não acontecia com a pequena esfera colorida, que existia em cada um deles. As crianças (e não só), munidas de algumas moedas, "faziam furos", pagando por cada um deles uma certa quantia. Pela parte da frente da cartolina, empurravam com um ponteiro ou lápis, sobre os furos pretendidos. Do outro lado, por cada furo, aparecia uma esfera, cuja cor definia o prémio a que tinham direito.

Vamos supor os seguintes códigos de cores:

Vermelho ? Caixa de chocolates grande
Azul ? Caixa de chocolates média
Verde ? Chocolate grande
Amarelo ? Chocolate médio
Castanho ? Chocolate (muito) pequeno.

Como é fácil de imaginar, a cor que saía com maior frequência era o castanho e, muito raramente, o vermelho... Já agora, como curiosidade, o aliciante para que o jogo se completasse era associar ao último furo, para além do prémio respectivo, o mostruário de prémios, com um exemplar de cada prémio.

O programa que se pretende desenvolver é uma recriação do *Jogo dos Furos*, mas do qual apenas se pretende uma pequena parte. Para simplificar, vamos supor que cada cartolina tem sempre 100 furos e que a posição dos prémios é aleatória. O número de bolas coloridas tem a seguinte distribuição: castanho ? 50%; amarelo ? 25%; verde ? 13%; azul ? 7%; vermelho ? as restantes.

Imaginar que relativamente a uma abstracção *jogo de furos*, existe o construtor

- ***faz-jogo-furos***, sem parâmetros, que devolve uma entidade do tipo cartolina com um jogo de 100 furos, preenchido nas proporções acima referidas.

3.1 Indicar a estrutura de dados que utilizaria para a entidade ***cartolina*** da abstracção do *jogo de furos*. Justificar a opção.

3.2 Escrever em *Scheme* o construtor ***faz-jogo-furos***, bem como os procedimentos auxiliares necessários.

(Fim.)