Prática de programação OCaml

Ficha de exercícios

Simão Melo de Sousa

Exercício 1 (Digressões sobre Fibonacci) *O objectivo deste exercício é escrever um programa Ocaml que permita calcular para um dado $n \in \mathbb{N}$ o valor de $Fib(n)$, tendo em conta que :*

$$\begin{cases} Fib(0) = 1 \\ Fib(1) = 1 \\ Fib(n+2) = Fib(n) + Fib(n+1) \end{cases}$$

1. *Numa primeira abordagem, escreva um programa que peça o valor de n via menu. O calculo deverá ser feito de forma recursiva;*

   Resposta

2. *modifique o seu programa de forma a que o valor de n seja dado pela linha de comando;*

   Resposta

3. *modifique o seu programa de forma a que o valor de n seja extraído de um ficheiro chamado* `dados.txt`*;* Resposta

4. *dê uma versão iterativa do calculo de $Fib(n)$;* Resposta

5. *dê uma versão recursiva terminal do calculo de $Fib(n)$;* Resposta

6. *dê uma versão com* memoization *da função $Fib(n)$;*

   Resposta

7. *considere a propriedade seguinte da sequência de fibonacci que comece em* 0 *(*$0, 1, 1, 2, 3, 5, 8, 13, \cdots$*):*

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n = \begin{bmatrix} Fib\ (n+1) & Fib\ n \\ Fib\ n & Fib\ (n-1) \end{bmatrix}$$

Tenha em consideração a propriedade seguinte da exponenciação de matrizes quadradas M por um inteiro n:

$$M^n = \begin{cases} M^{\frac{n}{2}} \times M^{\frac{n}{2}} & n \text{ par} \\ M \times M^{\frac{n}{2}} \times M^{\frac{n}{2}} & n \text{ impar} \end{cases}$$

Quando bem usada*, esta propriedade permite uma exponenciação em tempo logarítmico.*

Dê uma versão da função fibonacci que tire proveito desta propriedade;

Resposta

8. *o problema com a solução da alínea anterior é a conjunção da sua capacidade melhorada em calcular valores da função fibonacci com o uso do tipo inteiro* `int`*. Estes valores rapidamente ultrapassam a capacidade do tipo* `int`*.*

   Altere a resolução anterior para que se use o módulo `Num` *(módulo OCaml com tipos de dados e operações para a aritmética de precisão infinita).*

   Resposta

□

Exercício 2

(Cálculos sobre Polinómios de uma variável - Método de Horner e derivação)

Podemos representar um polinómio P de grau n por uma lista p de reais em o i-ésimo elemento da lista representa o coeficiente associado à potência de grau i. Assim, a título de exemplo, o polinómio $3x^4 + 5x^2 + 1$ é dado pela lista `[3;0;5;0;1]` *(ou* `[1;0;5;0;3]`*, se preferirmos que o polinómio seja listado do grau mais fraco ao grau mais forte).*

1. *Escreva uma função que peça o valor de n (com a restrição que $n \geq 0$) e inicialize P (lê os diferentes coeficientes a_i, onde $0 \leq i \leq n$).* Resposta

2. *Escreva uma função de escrita/visualização (output) que dado um polinómio p na sua forma de lista permite a sua visualização no* standard output*.* Resposta

3. *Escreva uma função que dado um x, calcule $P(x)$ usando o método de Horner, i.e.*

$$P_n(x) = (\cdots((a_n x + a_{n-1})x + a_{n-2})x + \cdots + a_1)x + a_0$$

   Resposta

4. *Escreva uma função que, dado um polinómio $P(x)$ dado na forma de uma lista, calcula o polinómio derivada de P em x.* Resposta

5. *Se a solução do ponto anterior não estiver numa forma recursiva terminal, proponha uma solução que o seja.* Resposta

□

Exercício 3 (listas e sub-listas)

1. *Define a função* `sublista l` *que calcula a lista de todas as sublistas de l com os elementos na ordem presente na lista original (ou seja* `[a;c]` *é sublista de* `[a;b;c]` *mas não* `[c;a]`*). Por exemplo:*

   ```
   sublista [a;b;c]= [[];[c];[b];[b;c];[a];[a;c];[a;b];[a;b;c]]
   ```

   Resposta

2. *Define a função* `insertion e l` *que calcula a lista de todas as listas possíveis que resultam da inserção de* `e` *em* `l`*. Por exemplo:*

   ```
   insertions e [a;b;c] = [[e;a;b;c];[a;e;b;c];[a;b;e;c];[a;b;c;e]]
   ```

   Resposta

3. *Define a função* `permutation l` *que calcula a lista de todas as permutações de l. Por exemplo:*

   ```
   permutation [a;b;c] = [[a;b;c];[b;a;c];[b;c;a];[a;c;b]; [c;a;b]; [c;b;a]]
   ```

   Resposta

4. *Define a função* `subbag l` *que calcula a lista de todas as permutações de todas as sublistas de l. Pretendemos gerar todas as sublistas possíveis duma determinada lista. Este exercício assemelha-se a geração do conjunto de todos os subconjuntos dum determinado conjunto com a particularidade de que a ordem dos elementos importa (a lista* `[a;b]` *é diferente da lista* `[b;a]` *e pretendemos aqui considera-las ambas). Por exemplo:*

   ```
   subbag [a;b;c] = [[]; [a]; [b]; [c]; [a;b]; [a;c] ; [b;a] ; [b;c] ; [c;a]
   ; [c;b] ; [a;b;c]; [a;c;b] ; [b;a;c] ; [b;c;a]; [c;b;a] ; [c;a;b]]
   ```

   Resposta

□

Exercício 4 (Códigos de Gray) *Os códigos de Gray permitam uma codificação binária cómoda em que um só bit difere entre elementos consecutivos. Para simplificar, iremos nos restringir ao caso dos inteiros. Neste caso, a codificação de* 0 *é* 0 *e a codificação de* 1 *é* 1*. A codificação de* 17 *é* 11001*, a codificação de 18 é* 11011 *e a codificação de* 19 *é* 11010*.*

Uma forma simples de gerar os códigos de gray dos valores inteiros até ao tamanho n (por exemplo 19 *tem uma codificação de tamanho* 5*) é designada de método* reflex-and-prefix*.*

Este método é explicado pela imagem da figura 1 (fonte : wikipedia)

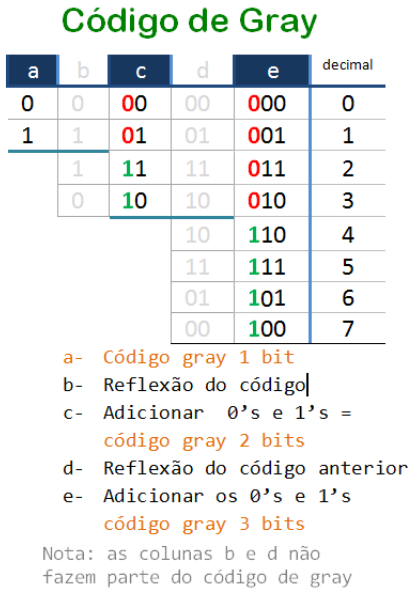

Figura 1: Método reflex and prefix (fonte : wikipedia)

1. *Define uma função que dado um n calcula todos os códigos de gray de tamanho n. Estes código são devolvidos na forma de uma lista;* Resposta

2. *define uma função que dê o código de gray de um determinado inteiro n em parâmetro.* Resposta

□

Exercício 5 (Árvores AVL) *Neste exercício vamos re-descobrir uma estrutura de dados clássica, as árvores AVL que são árvores binárias ordenadas e equilibradas.*

1. *Define o tipo (polimórfico) das árvores binárias eventualmente vazias com a informação da altura em cada nodo (por exemplo uma árvore de altura x tem esta informação arquivada na raiz);* Resposta

2. *define igualmente a função* `map f ar` *sobre as árvores;* Resposta

3. *define a função* `fold_bfs f init ar` *(breadth-first search, em largura primeiro, da esquerda para a diteita) que aplica a função binária* `f` *a todos os elementos da árvore* `ar`*. O valor inicial do acumulador é* `init`*;* Resposta

4. *proponha uma versão recursiva terminal de* `fold_dfs` *e de* `fold_bfs`*;* Resposta

5. *define uma função* `insert e ar` *que insere o elemento* `e` *ordenadamente e de forma equilibrada na árvore* `ar`*;* Resposta

6. *define uma função* `remove e ar` *que remove o elemento* `e` *ordenadamente e de forma equilibrada da árvore* `ar`*.* Resposta

□

Exercício 6 (Totoloto) *Neste exercício vamos simular um sorteio do totoloto.*

1. *Codificar a grelha como uma matriz* 7×7 *de booleanos;*

2. *definir uma função que preencha a grelha a partir de uma sequência de* 7 *números distintos (entre* 1 *e* 49*);*

3. *definir uma função que dado um sorteio (lista de* 6 *números inteiros complementado com um inteiro – o complementar) diz em quantos numeros se acertou.*

Resposta □

Exercício 7 (FNC)
O objectivo deste exercício é a exploração dos tipos soma. No caso particular deste exercício do tipo das expressões lógicas (proposicionais).

1. *Define o tipo das expressões lógicas proposicionais;* Resposta

2. *define o tipo das formas normais conjuntivas (FNC);* Resposta

3. *implemente o algoritmo* $\mathcal{T}$ *que permite a conversão de uma fórmula lógica proposicional para FNC;* Resposta

□