Linguagens e Ambientes de Programação

Exame de Época Normal 2010/2011 – Proposta de Resolução

3.

```
(*0 preto*)
(*1 branco*)

type quadtree = QColor of int | QNode of quadtree * quadtree * quadtree *
quadtree;;

let quadExample = QNode(QColor 0, QNode( QColor 0, QColor 0, QColor 1, QColor
1), QNode( QColor 0, QNode( QColor 0, QColor 1, QColor 1, QColor 1), QColor 0,
QColor 1), QNode( QColor 1, QColor 1, QNode( QColor 1, QColor 1, QColor 1,
QColor 0), QColor 0));;

let listExample = ['Q'; '0'; 'Q'; '0'; '0'; '1'; '1'; 'Q'; '0'; 'Q'; '0'; '1';
'1'; '1'; '0'; '1'; 'Q'; '1'; '1'; 'Q'; '1'; '1'; '1'; '0'; '0'];;

let rec count q =
      match q with
            QColor _ -> 1
      |     QNode(a,b,c,d) -> count a + count b + count c + count d
;;
(*count example;;*)

let rec whiteness q =
      match q with
              QColor 1 -> 1.0
            | QColor _ -> 0.0
            | QNode(a,b,c,d) ->
            (whiteness a +. whiteness b +. whiteness c +. whiteness d) /. 4.0
;;
(*whiteness example;;
whiteness (QColor 0);;
whiteness (QColor 1);;*)

let rec qor q1 q2 =
      match q1, q2 with
              QColor c1, QColor c2                  -> QColor (c1 lor c2)
            | QColor 1, QNode(a,b,c,d)              -> QColor 1
            | QColor _, QNode(a,b,c,d)              -> q2
            | QNode(a,b,c,d) ,QColor 0              -> q1
            | QNode(a,b,c,d) ,QColor _              -> QColor 1
            | QNode(a1,b1,c1,d1) ,QNode(a2,b2,c2,d2) ->
              let s = QNode(qor a1 a2, qor b1 b2, qor c1 c2, qor d1 d2) in
                  let w = whiteness s in
                        if (w <> 0.0 && w <> 1.0) then s
                        else QColor (int_of_float w)
;;
(*whiteness(QNode(QColor 1, QColor 1,QColor 0,QColor 1));;
qor (QNode(QColor 1, QColor 0,QColor 1,QColor 0)) example;;*)
```

```
let rec tolist q =
      match q with
              QColor 0 -> ['0']
            | QColor _ -> ['1']
            | QNode(a,b,c,d) -> 'Q'::(tolist a @ tolist b @ tolist c @ tolist d)
;;
(*tolist example;;*)

let rec fromlistX l =
      match l with
              [] -> (QColor 0, ['0']) (*PARA EVITAR WARNINGS*)
            |     '0'::xs ->  (QColor 0, xs)
            | '1'::xs ->  (QColor 1, xs)
            | x::xs   ->
              let (nw, brothers1) = fromlistX xs in
                  let (ne, brothers2) = fromlistX brothers1 in
                        let (sw, brothers3) = fromlistX brothers2 in
                              let (se, brothers4) = fromlistX brothers3 in
                                    (QNode(nw, ne, sw, se), brothers4)
;;
let fromlist l = fst (fromlistX l) ;;
(*fromlist ['Q';'0';'1';'1';'Q';'1';'1';'1';'0'];;*)
```

5.

```c
#include <stdlib.h>
#include <stdio.h>
typedef double Data;
typedef struct Node{
	Data data;
	struct Node *next;
}Node, *List;

typedef struct DNode{
	Data data;
	struct DNode *prev, *next;
}DNode, *DList;

List NewNode(Data val, List next){
	List l = malloc(sizeof(Node));
	if( l == NULL ) return NULL;
	l->data = val;
	l->next = next;
	return l;
}

DList NewDNode(Data val, DList prev, DList next){
	DList n = malloc(sizeof(DNode));
	if( n == NULL )return NULL;
	n->data = val;
	n->prev = prev;
	n->next = next;
	return n;
}

DList List2DList(List l){
	DList n = NewDNode(l->data, NULL, NULL);
	DList dl = n;
	while(l->next){
		l = l->next;
		n->next = NewDNode(l->data, n, NULL);
		n = n->next;
	}
	return dl;
}

void printDList(DList d){
	while(d->next){
		printf("%g ",d->data);
		d = d -> next;
	}
	printf("%g \n",d->data);
	while(d){
		printf("%g ",d->data);
		d = d->prev;
	}
	printf("\n");
}

int main(void) {
	List l = NewNode(1, NewNode(2, NewNode(3, NewNode(4, NewNode(5,
			NewNode(6, NewNode(7, NewNode(8, NewNode(9, NewNode(10,
			NULL)))))))))));
	printDList(List2DList(l));
	return 0;
}
```