Fondamenti di Informatica T-1 (A.A. 2018/2019) - Ingegneria Informatica

Prof.ssa Mello

Prova Scritta – 24 Gennaio 2019 – durata 1h

Totale 12 punti, sufficienza con 7

Compito A

ESERCIZIO 1 (6 punti)

Si scriva una funzione RICORSIVA `elemSum`

```
list elemSum(list l, int elem)
```

che, data in ingresso una lista di interi `l`, fornisca in uscita una nuova lista contenente gli elementi della lista data in ingresso, a cui però sia stato sommato (ad ogni elemento) il valore `elem`.

Si realizzi inoltre una funzione ITERATIVA `lastList`

```
int lastList(list l)
```

che ritorni l'ultimo elemento di `l`, supponendo che la lista di partenza contenga almeno un elemento.

Si realizzi infine una funzione `main()` che utilizzi correttamente tali funzioni per stampare a terminale una lista `l2` ottenuta sommando a ogni elemento della lista `l1=[2,0,5,-1,3]` il valore dell'ultimo elemento di `l1`. (Nota: dovrà essere quindi `l2=[5,3,8,2,6]`).

È possibile avvalersi di funzioni di supporto purché si preservi la struttura ricorsiva o iterativa richiesta: tali funzioni andranno ovviamente riportate. **Tutte le funzioni dovranno essere implementate utilizzando le primitive dell'ADT lista, includendo nel main l'header `"list.h"`. Non è necessario riportare la dichiarazione (list.h) e l'implementazione delle primitive (list.c).**

ESERCIZIO 2 (2 punti)

Si consideri la seguente funzione

```
int s(int n, int a) {
      if ( n <= 0 ) {
            return a;
      } else {
            return s(n-1, a+n) + n-1;
      }
}
```

Mostrare la sequenza dei record di attivazione ed il valore di ritorno nel caso in cui la funzione sia invocata con parametri attuali (3, 0).

ESERCIZIO 3 (3 punti)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data).

```
#include <stdlib.h>
#include <stdio.h>

int* pDet(int m[][2], int s) {
    int i, a = m[0][0] * m[1][1] - m[0][1] * m[1][0];
    int *res = (int*) malloc(sizeof(int)*s);
    for (i = 0; i < s; ++i,a*=2) {
        res[i] = a;
    }
    return res;
}

int* bpDet(int m[][2], int n) {
    int i, j;
    int *b = pDet(m,n*2-1);
    int *res = (int*) malloc(sizeof(int)*n);
    for (i = 0, j = 0; i < n; ++i, j+=2) {
        res[i] = b[j];
    }
    free(b);
    return res;
}

int main() {
    int n = 3, i, j, v = 0, m[2][2];
    int *res;
    for (i = 0; i < 2; ++i) {
        for (j = 0; j < 2; ++j) {
            m[i][j] = v++;
        }
    }
    res = bpDet(m,n);

    for (i = 0; i < n; i++) {
        printf("%d ", res[i]);
    }

    free(res);
    return 0;
}
```

ESERCIZIO 4 (1 punto)

Si descrivano brevemente le differenze tra il passaggio dei parametri per valore e per riferimento.

Soluzioni

ESERCIZIO 1

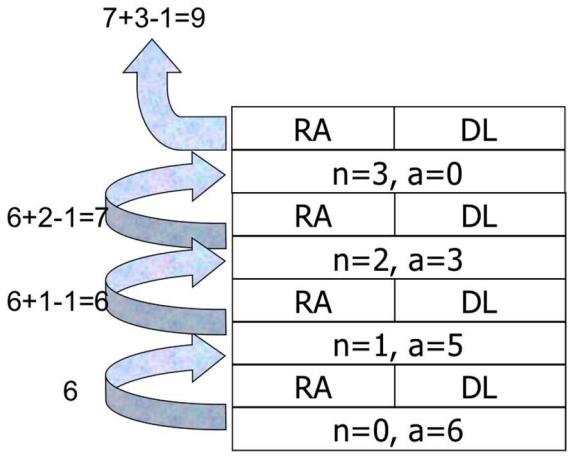
```
#include <stdlib.h>
#include <stdio.h>
#include "list.h"

list elemSum(list l, int sum) {
    if (empty(l)) {
        return emptyList();
    } else {
        return cons(sum + head(l), elemSum(tail(l),sum));
    }
}

int lastList(list l) {
    int res;
    while (!empty(l)) {
        res = head(l);
        l= tail(l);
    }
    return res;
}

int main() {
    list l2, l1 = cons(2,cons(0,cons(5,cons(-1,cons(3,emptyList())))));
    l2 = elemSum(l1,lastList(l1));
    while(!empty(l2) ){
        printf("%d\n", head(l2));
        l2 = tail(l2);
    }
    return 0;
}
```

ESERCIZIO 2



ESERCIZIO 3

Il programma è corretto e l'output prodotto è:

```
-2
-8
-32
```

La funzione `main` inizializza `n = 3`, crea una matrice `m` 2x2 e la inizializza con gli interi da 0 a 3 e chiama `bpDet` con parametri `m` e `n`.

`bpDet` chiama `pDet` con parametri `m` e `n*2-1`, ossia 5.

`pDet` assegna all'intero `a` il risultato di un'espressione composta da 2 moltiplicazioni e una sottrazione la quale calcola il determinante di una matrice 2x2, che in questo caso è 0*3-1*2 = -2. Successivamente `pDet` alloca un array di interi res di `s = 5` elementi. Tramite un ciclo for che moltiplica per 2 ad ogni iterazione inizializza l'array con `[-2,-4,-8,-16,-32]` e lo ritorna.

`bpDet` assegna al puntatore `b` il risultato di `pDet`, e alloca un array `res` di interi con dimensione `n = 3`. Con un ciclo for assegna agli elementi di `res` gli elementi d'indice pari (posizioni 0, 2 e 4) di `b`, visto come array. Si noti infatti che l'indice `j` è incrementato di due per ogni ciclo. `res` contiene dunque `[-2, -8, -32]`. La memoria di `b` viene liberata e `res` ritornato.

Il `main` stampa il risultato di `bpDet`, libera la memoria dell'array risultato e termina.