Fondamenti di Informatica T-1 (A.A. 2015/2016) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Giovedì 28 Gennaio 2016 – durata 1h
Totale 12 punti, sufficienza con 7

Compito A

ESERCIZIO 1 (6 punti)

Sia data una lista di interi `l`. Si realizzi la funzione ITERATIVA

```
list invert(list l);
```

che inverta l'ordine degli elementi in una lista.

Siano date due liste di interi contenenti lo stesso numero di elementi `l1` e `l2`. Si realizzi la funzione RICORSIVA

```
list sum(list l1, list l2);
```

che ritorni una lista di interi contenente lo stesso numero di elementi di `l1` e `l2` dove gli elementi alla stessa posizione siano sommati.

Si realizzi infine una funzione `main()` che crei due liste di interi di uguale numero di elementi `l1` e `l2` ed invochi correttamente le funzioni precedenti per calcolare una lista di interi contenente lo stesso numero di elementi di `l1` e `l2` dove gli elementi alla stessa posizione siano sommati, ma considerando gli elementi di `l2` in ordine inverso. Per esempio, se `l1={45,-3,81,-34,0}` e `l2={18,12,-2,-14,31}`, la lista calcolata sarà `{76,-17,79,-22,18}`.
Le funzioni dovranno essere implementate utilizzando le primitive dell'ADT lista, includendo `"list.h"`.

ESERCIZIO 2 (2 punti)

Un elaboratore rappresenta i numeri interi su 8 bit tramite la notazione in complemento a 2. Indicare come viene svolta la seguente operazione aritmetica calcolandone il risultato secondo la rappresentazione binaria in complemento a 2 (si trasli anche il risultato in decimale per verificare la correttezza dell'operazione):
12 – 23

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

int op(int n1, int n2, char c) {
    switch (c) {
        case 'M':
            return n1 > n2 ? n1 : n2;
        case 'm':
            return n1 < n2 ? n1 : n2;
        case 's':
```

```
            return n1 + n2;
        case 'p':
            return n1 * n2;
        default:
            abort();
    }
}

int opEach(int a[], int l, char c) {
    int res = a[0];
    int i;
    for (i = 1; i < l; ++i) {
        res = op(res,a[i],c);
    }
    return res;
}

int main(){
    int a[] = {4,6,0,8,-12,21,10};
    printf("%d\n",opEach(a,7,'M'));
    return 0;
}
```

ESERCIZIO 4 (1 punto)

Si illustri brevemente il funzionamento dell'algoritmo bubblesort e le sue caratteristiche.

Soluzioni

ESERCIZIO 1

Nota: le funzioni `printElement`, `printList` e loro chiamate non sono strettamente richieste dall'esercizio.

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

void printElement(element e) {
    printf("%d",e);
}

void printList(list l) {
    printf("[");
    while (empty(l)==false) {
        printElement(head(l));
        printf(" ");
        l = tail(l);
    }
    printf("]");
}

list invert(list l) {
    list res = emptyList();
    while(empty(l) == false){
        res = cons(head(l),res);
        l = tail(l);
    }
    return res;
}

list sum(list l1, list l2) {
    if (empty(l1) == true) {
        return emptyList();
    } else {
        return cons(head(l1)+head(l2),sum(tail(l1),tail(l2)));
    }
}

int main() {
    list l1 = emptyList();
    list l2 = emptyList();
    l1 = cons(0,l1);
    l1 = cons(-34,l1);
    l1 = cons(81,l1);
    l1 = cons(-3,l1);
    l1 = cons(45,l1);
    l2 = cons(31,l2);
    l2 = cons(-14,l2);
    l2 = cons(-2,l2);
    l2 = cons(12,l2);
    l2 = cons(18,l2);
    printList(l1);
    printList(l2);
    printList(sum(l1,invert(l2)));
    return(0);
```

```
}
```

ESERCIZIO 2

12 - 23 = - 11

12 = 8 + 4 → 00001100
23 = 16 + 4 + 2 + 1 → 00010111
- 23 → 11101001
12 – 23 → 11110101
- 11110101 = 00001011 → 11

ESERCIZIO 3

Il programma è corretto sintatticamente e la sua esecuzione produce la stampa:

```
21
```

La funzione `op()` ha come parametri due interi `n1` e `n2` ed un carattere `c`. Secondo il valore assunto da `c` esegue una delle seguenti operazioni.

- Se `c=='M'` ritorna il massimo tra i due interi.
- Se `c=='m'` ritorna il minimo tra i due interi.
- Se `c=='s'` ritorna la somma dei due interi.
- Se `c=='p'` ritorna il prodotto dei due interi.

La funzione `opEach()` ha come parametri un array `a`, un intero `l` e un carattere `c`. Se `l` è uguale a `1` ritorna l'elemento in posizione `0` di `a`. In caso contrario ritorna il risultato della funzione `op()` applicata iterativamente a tutti gli elementi di `a`.

- Se `c=='M'` ritorna il massimo tra tutti gli elementi.
- Se `c=='m'` ritorna il minimo tra tutti gli elementi.
- Se `c=='s'` ritorna la sommatoria.
- Se `c=='p'` ritorna la produttoria.

La funzione `main()` inizializza un array `a`, chiama la funzione `opEach()` con parametri `a`, `7`, '`M`' e ne stampa il risultato. A seconda del carattere scelto, `opEach()` ritorna il massimo, il minimo, la sommatoria o la produttoria dell'array passato. L'intero fornito a `opEach()` come secondo parametro è la dimensione logica dell'array.