

Fondamenti di Informatica T-1 (A.A. 2016/2017) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Giovedì 12 Gennaio 2017 – durata 1h
Totale 12 punti, sufficienza con 7

Compito A

ESERCIZIO 1 (6 punti)

Siano date due liste `l1` e `l2` di interi. Si assuma che le due liste abbiano pari lunghezza e che gli interi memorizzati siano strettamente positivi. Si realizzi la funzione RICORSIVA

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

che ritorni una nuova lista `l3` avente la stessa lunghezza di `l1` e `l2` dove l'elemento `k`-esimo vale 1 se l'elemento `k`-esimo di `l1` è un multiplo dell'elemento `k`-esimo di `l2`, e 0 altrimenti.

Si implementi inoltre una funzione ITERATIVA:

```
int conta(list l3)
```

che conti il numero di multipli trovati, ovvero il numero di valori pari a 1 nella lista `l3`.

Si realizzi infine una funzione `main()` che crei le due liste `l1` e `l2` ed utilizzi correttamente le due funzioni `verificaMultipli(l1, l2)` e `conta(l3)` precedenti in modo tale da calcolare la somma finale. Ad esempio, date le liste `l1 = {16, 23, 8, 10, 2, 44}` e `l2 = {8, 12, 8, 3, 4, 11}` si dovrà ottenere la lista `l3 = {1, 0, 1, 0, 0, 1}` e la somma finale sarà pari a 3.

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):
4 – 11

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 f(char* x, int *y){
    int i=0, j=1;
    do
    {
        if (x[i++] == 'a')
        {
            (*x)++;
        }
        else
        {
            i--;
        }
        printf("%c %c\n",x[i],x[j--]);
    } while (i>=0 && j>=0);
    *y=++i;
    return j;
}

int main(){
    char x[]={ 'a', 'e', 'i', 'o', 'u' };
    int y=0;
    int res=f(x,&y);
    printf("%d %d\n",y,res);
    return 0;
}
```

ESERCIZIO 4 (1 punto)

Si introduca il concetto di record d'attivazione di una funzione e se ne spieghi la struttura, anche con l'aiuto di esempi.

Soluzioni

ESERCIZIO 1

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

list verificaMultipli(list l1, list l2)
{
    list l3 = emptyList();

    if (empty(l1))
    {
        return emptyList();
    }
    else
    {
        if (head(l1) % head(l2) == 0)
            return cons(1,verificaMultipli(tail(l1),tail(l2)));
        else
            return cons(0,verificaMultipli(tail(l1),tail(l2)));
    }
}

int conta(list l3)
{
    int somma = 0;
    while (l3 != NULL)
    {
        somma += head(l3);
        l3 = tail(l3);
    }
    return somma;
}

int main()
{
    list L1 = emptyList();
    list L2 = emptyList();
    list L3 = emptyList();
    list tmp;

    L1 = cons(16,L1);
    L1 = cons(23,L1);
    L1 = cons(8,L1);
    L1 = cons(10,L1);
    L1 = cons(2,L1);
    L1 = cons(44,L1);

    L1 = cons(8,L1);
    L1 = cons(12,L1);
    L1 = cons(8,L1);
    L1 = cons(3,L1);
    L1 = cons(4,L1);
}
```

```

L1 = cons(11,L1);

L3 = verificaMultipli(L1,L2);
printf("%d",conta(L3));

showList(L3);

while(!empty(L1)) { tmp = L1; L1 = tail(L1); free(tmp); }
while(!empty(L2)) { tmp = L2; L2 = tail(L2); free(tmp); }
while(!empty(L3)) { tmp = L3; L3 = tail(L3); free(tmp); }

return 0;
}

```

ESERCIZIO 2

4 - 11 = - 7

4 → 00000100

11 = 8 + 2 + 1 → 00001011

- 11 → 11110101 (rappresentazione in complemento a 2)

4 - 11 → 11111001

11111001 (rappresentazione in complemento a 2) = -7

ESERCIZIO 3

L'output prodotto è

```

e e
e b
2 -1

```

Il programma `main` crea un array `x` di caratteri ed invoca la funzione `f`, utilizzando come parametri l'array stesso e la variabile `y=0` (passata per riferimento). Il risultato è memorizzato in `res`.

La funzione `f` esegue una prima volta il ciclo `do-while` ed effettua un confronto tra la variabile `x[0]` (poiché viene usato il post-incremento sulla variabile `i`) e il carattere `'a'`. La condizione è quindi verificata, per cui viene eseguita l'istruzione `(*x)++` che incrementa il contenuto della variabile puntata da `x`, modificandone quindi il valore da `'a'` a `'b'`. La funzione quindi esegue il `printf` della variabile `x[1]` due volte, poiché `i` è stata incrementata in precedenza e vale 1, mentre `j` vale 1 prima del post-incremento.

Viene quindi eseguita una seconda iterazione del ciclo, dove il primo confronto fallisce (in quanto `x[1]` è diverso da `'a'`), ma viene post-incrementato il valore di `i`, che a questo punto vale 2. Viene quindi eseguito il corpo dell'istruzione `else` decrementando il valore di `i`, che torna quindi ad assumere valore 1. A questo punto la `printf` stampa `x[1]`, che vale `'e'` e `x[0]` che vale `'b'`. Dopo il post-incremento, la variabile `j` vale -1 per cui il ciclo termina.

Una volta terminato il ciclo, viene assegnato il valore `++i`, quindi 2 (pre-incremento), al contenuto della variabile puntata dal puntatore `y`. Infine, la funzione ritorna il valore `j` ovvero -1.

Nel `main` vengono stampati i valori di `y` (modificato dalla funzione, quindi 2) e `res` (ovvero -1).