

Fondamenti di Informatica T-1 (A.A. 2013/2014) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Giovedì 30 Gennaio 2014 – durata 1h
Totale 12 punti, sufficienza con 7

Compito A

ESERCIZIO 1 (6 punti)

I numeri immaginari sono composti da una parte reale e una parte complessa (in questo caso due numeri interi). Si faccia riferimento alla seguente struttura dati:

```
typedef struct {
    int real;
    int complex;
} Imaginary;
```

Due numeri complessi che hanno stessa parte reale e parte complessa opposta si dicono coniugati, ad esempio $A=(2,5)$ e $B=(2,-5)$, e la loro somma produce un numero complesso C che ha come parte reale la somma delle parti reali di A e B (cioè $2+2=4$) e parte complessa nulla: $C=(4,0)$.

Date due liste di numeri immaginari $l1$ e $l2$ con lo stesso numero di elementi, si realizzi una funzione RICORSIVA

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

che inserisca nella lista restituita la somma tra un numero immaginario contenuto nella prima lista e il corrispondente coniugato contenuto nella seconda lista. Se, invece, un elemento di $l1$ non ha un corrispondente coniugato in $l2$, si saltino entrambi gli elementi senza inserire alcuna somma nella lista. Per esempio, se $l1=[(2,5), (2,-1), (3,-7)]$ e $l2=[(2,-5), (4,0), (3,7)]$, la funzione `sum_complex()` deve restituire la lista $[(4,0), (6,0)]$. In particolare, il secondo elemento $(6,0)$ è incluso nella lista restituita, in quanto la somma dei numeri complessi coniugati $(3,7)$ e $(3,-7)$ risulta essere pari a $(6,0)$, mentre l'elemento $(2,-1)$ contenuto in $l1$ e l'elemento corrispondente $(4,0)$ in $l2$, non saranno inclusi poiché non sono coniugati.

Si definisca inoltre la funzione

```
int compareConjugate (Imaginary e11, Imaginary e12)
```

che dati in ingresso due numeri immaginari, ne faccia il confronto stabilendo se sono o meno coniugati.

La funzione `sum_complex()` dovrà essere implementata utilizzando le sole primitive dell'ADT lista; ogni altra funzione dovrà essere opportunamente specificata dal candidato. Si realizzi inoltre una semplice funzione `main()` di prova che invochi correttamente la funzione `sum_complex()` creata. A tal fine, si consideri come nota la funzione `Imaginary create(int real, int complex)` che presi in ingresso due numeri interi, restituisca un numero immaginario. Ad esempio, per creare la lista $[(1,3)]$: `list l = cons(create(1,3), emptylist());`

ESERCIZIO 2 (2 punti)

Si consideri la seguente grammatica G con scopo S e simboli terminali $\{a, b, c, +, *\}$

```
S ::= E
E ::= F | GT | FE
T ::= EG | E
F ::= a | b | c
G ::= + | *
```

La stringa $"b+a*b^*"$ appartiene al linguaggio di tale grammatica? In caso affermativo se ne mostri la derivazione left-most.

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>

char* fun(char *a){
    int i, k, length=0;
    char* res = NULL;
    char* c = a;

    while(*a != '\0'){
        a++;
        length++;
    }

    res = (char*)malloc(sizeof(char)*(length+1));

    for(k=length-1; k>=length/2; k--){
        res[k] = *(c+k)-1;
    }

    for(i=0; i<length/2; i++){
        res[i]= *c+i*3;
    }
    res[length] = '\0';

    return res;
}

int main(void){
    char *str;
    char v1[] = {'a','b','c','d','e','f','\0'};

    str = fun(v1);
    while(*str!='\0'){
        printf("%c",*str);
        str++;
    }

    return 0;
}
```

ESERCIZIO 4 (1 punto)

Si descrivano brevemente le differenze principali tra variabili locali e globali, fornendo anche semplici esempi di codice in linguaggio C.

Soluzioni

ESERCIZIO 1

```
int compareConjugate (Imaginary e1, Imaginary e2){
    if(e1.real == e2.real && e1.complex == (-1 * e2.complex))
        return 1;
    else
        return 0;
}

list sum_complex(list l1, list l2) {
    Imaginary tmp;

    if(empty(l1))
        return l1;
    else{
        if(!compareConjugate(head(l1),head(l2)))
            return sum_complex(tail(l1),tail(l2));
        else{
            tmp.real = head(l1).real*2;
            tmp.complex = 0;
            return cons(tmp, sum_complex(tail(l1),tail(l2)));
        }
    }
}

int main(void)
{
    list l1, l2, res;

    l1 = cons(create(2,5), cons(create(2,-1), cons(create(3,-7), emptylist())));
    l2 = cons(create(2,-5), cons(create(4,0), cons(create(3,7), emptylist())));

    printf("sum_complex\n");
    res = sum_complex(l1,l2);
    while( !empty(res) ){
        printf("(%d,%d)\n", head(res).real, head(res).complex);
        res = tail(res);
    }

    return (0);
}
```

ESERCIZIO 2

La frase appartiene al linguaggio. In particolare, la si può ottenere tramite la seguente derivazione left-most:

$S \rightarrow E \rightarrow FE \rightarrow bE \rightarrow bGT \rightarrow b+T \rightarrow b+EG \rightarrow b+FEG \rightarrow b+aEG \rightarrow b+aGTG \rightarrow b+a^*TG \rightarrow b+a^*EG \rightarrow b+a^*FG \rightarrow b+a^*bG \rightarrow b+a^*b^*$

ESERCIZIO 3

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

adgcde

La funzione `main()` invoca la funzione `fun()` passando come parametri di ingresso la stringa ben formata `v1`.

La funzione iterativa `fun()` incrementa il valore del puntatore `a` e della variabile intera `length` fino al terminatore di stringa; poi alloca dinamicamente spazio di memoria equivalente alla memoria occupata da `a` (incluso il terminatore di stringa). Il primo ciclo `for` inserisce nella seconda metà dell'area di memoria allocata dinamicamente i valori dei caratteri corrispondenti della stringa ben formata `c`, ai quali è sottratto 1, partendo dall'ultimo carattere. Il secondo ciclo `for`, inserisce i valori mancanti nell'area di memoria allocata dinamicamente, questa volta attraverso un ciclo in avanti a partire dalla prima posizione; ad ogni iterazione si considera il valore della posizione iniziale della stringa ben formata `c` (il carattere `'a'`), al quale è aggiunto il triplo dell'indice `i` del ciclo `for`. Infine la funzione `fun()` inserisce il terminatore di stringa (ovvero il valore 0) in fondo all'area di memoria allocata dinamicamente, di cui restituisce un riferimento.

La funzione `main()` stampa sullo standard output i valori presenti nell'area di memoria allocata dinamicamente, fino al terminatore escluso.