Fondamenti di Informatica T-1 (A.A. 2017/2018) - Ingegneria Informatica
Prof.ssa Mello
Prova Scritta - 12 Luglio 2018 – durata 1h
Totale 12 punti, sufficienza con 7

ESERCIZIO 1 (6 punti)

Si scriva una funzione RICORSIVA `delta`:

```
list delta(list list1);
```

che data una lista di interi `list1`, restituisca una nuova lista contenente le differenze tra i numeri contigui, che risultano sottraendo il primo elemento al secondo, poi il secondo al terzo ecc. Qualora la lista `list1` risulti essere vuota, o contenere un solo elemento, la funzione restituisca in uscita una lista vuota.

Ad esempio se `list1 = [4, 5, 1, 7, 11]`, la funzione `delta` deve restituire la lista `[1, -4, 6, 4]`. Si noti che la lista risultante ha un elemento in meno rispetto a quello di partenza e l'ordine è mantenuto.

La funzione dovrà essere implementata utilizzando le sole primitive dell'ADT lista, includendo "list.h". Ogni altra funzione dovrà essere codificata dal candidato. Si realizzi inoltre una semplice funzione `main()` di prova che invochi correttamente la funzione `delta`.

ESERCIZIO 2 (2 punti)

Si consideri la seguente grammatica `G` con scopo `S`, simboli non terminali `{T, F}` e simboli terminali `{1, 0, -, x, <, >}`.

```
S := T
T := <TxT> | <TxF> | <FxT> | <1-F> | 1
F := <FxF> | <1-T> | 0
```

La stringa "<1-<<0x0>x<1-1>>>" 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 <stdlib.h>
#include <stdio.h>

int h(char* w, int s) {
    if (s > 1) {
        int i = h(w+1,s-1)+1;
        if (w[0] >= w[i]) {
            return 0;
        } else {
            return i;
        }
    } else {
        return 0;
    }
}

int main() {
    char w1[] = "an", w2[] = "to", w3[] = "all";
    printf("%c %c %c",w1[h(w1,2)],w2[h(w2,2)],w3[h(w3,3)]);
    return 0;
}
```

ESERCIZIO 4 (1 punto)

S'illustri il concetto di aritmetica dei puntatori e se ne mostri un semplice esempio.

Soluzioni

ESERCIZIO 1 soluzione 1

```
list delta(list l) {
    if (empty(l)) {
        return emptyList();
    } else {
        if (empty(tail(l))) {
            return emptyList();
        } else {
            return cons(head(tail(l))- head(l),delta(tail(l)));
        }
    }
}
```

ESERCIZIO 1 soluzione 2

```
list innerDelta(list l, int p) {
    if (empty(l)) {
        return emptyList();
    } else {
        int h = head(l);
        return cons(h-p,innerDelta(tail(l), h));
    }
}

list delta(list l) {
    if (empty(l)) {
        return emptyList();
    } else {
        return innerDelta(tail(l), head(l));
    }
}

int main() {
    list list1 = cons(4,cons(5,cons(1,cons(7,cons(11,emptyList())))));
    list list2 = delta(list1);
    while( ! empty(list1) ){
      printf("%d ", head(list1));
      list1 = tail(list1);
    }
    printf("\n");
    while( ! empty(list2) ){
      printf("%d ", head(list2));
      list2 = tail(list2);
    }
    return 0;
}
```

ESERCIZIO 2

La frase appartiene al linguaggio. In particolare, la si può ottenere tramite la seguente derivazione left-most:
S → T → <1-F> → <1-<FxF>> → <1-<<FxF>xF>> → <1-<<0xF>xF>> →
<1-<<0x0>xF>> → <1-<<0x0>x<1-T>>> → <1-<<0x0>x<1-1>>>

ESERCIZIO 3

Il programma compila. L'output prodotto è

```
n t l
```

Il programma inizializza tre array di caratteri `w1`, `w2` e `w3` con le stringhe "`an`", "`to`" e "`all`", poi invoca una `printf` che stampa tre caratteri, che sono i caratteri degli array corrispondenti alle posizioni ritornate da `h`, chiamata sugli stessi tre array di caratteri, dove il primo parametro è appunto l'array, mentre il secondo è `2`, `2` e `3`, ossia la lunghezza dell'array corrispondente.

La funzione `h` prende in ingresso un puntatore a carattere `w` ed un intero `s` e ritorna un intero. Se `s` non è maggiore di `1` ritorna `0`. Se s è maggiore di `1` calcola un intero `i` invocando ricorsivamente `h`, usando come parametri il puntatore al carattere successivo a `w` e `s-1`, e sommando `1` al valore ritornato da `h`. Poi se il carattere puntato da `w` è maggiore o uguale al carattere che lo segue di `i` posizioni, `h` ritorna `0`. In caso contrario ritorna `i`.

Si evince che la funzione `h` calcola la prima posizione dove compare il carattere maggiore (in senso alfabetico). Ottiene questo con una strategia ricorsiva, dove ad ogni chiamata confronta un carattere con il carattere più alto tra quelli che lo seguono, e ritorna la posizione del carattere maggiore, considerando il primo tra questi in caso di parità.

Visto che il `main` stampa i caratteri alle posizioni ritornate da `h`, i tre caratteri stampati sono quelli di valore più alto nelle stringhe "`an`", "`to`" e "`all`".