Fondamenti di Informatica T-1 (A.A. 2017/2018) - Ingegneria Informatica
Prof.ssa Mello
Prova Scritta - 25 Gennaio 2018 – durata 1h
Totale 12 punti, sufficienza con 7

Compito A

ESERCIZIO 1 (6 punti)
Data una lista `list1`, che contiene un numero pari di caratteri (o è vuota), si realizzi una funzione RICORSIVA

```
list coppie(list list1);
```

che restituisca una nuova lista contenente solo gli elementi `element1` di `list1` in posizione dispari a partire dal primo elemento, in modo che selezionando in essa a coppie gli elementi `element1 element2` a partire dalla testa della lista `list1`, rispettino la proprietà che `element1` preceda `element2` in ordine alfabetico o siano uguali.

Ad esempio se `list1 = ['c', 'a', 't', 'z', 'a', 'a']`, la funzione `coppie` applicata a tale lista deve restituire la nuova lista `['t', 'a']` che non contiene `'c'` (poiché non precede `'a'`) mentre contiene `'t'` perché precede `'z'`, e `'a'` perché è uguale a `'a'`).

La funzione 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 `coppie` su una lista composta da `['g', 'r', 'o', 'k']`e la stampi a terminale. Si scrivano le direttive di inclusione necessarie.

ESERCIZIO 2 (2 punti)

Si consideri la seguente grammatica `G` con scopo `S`, simboli non terminali `{B, N, P}` e simboli terminali `{1, 0, -, (, )}`. Si noti che anche le parentesi tonde sono considerate simboli terminali.

```
S := B
B := P | N
N := 0 | 1 | 0N | 1N
P := (N-B-B)
```

La stringa "**(10-1-(11-0-1))**" 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>
#include <string.h>

typedef struct Serie {
    char* titolo;
    int nBook;
} Serie;

Serie* initSerie(int nBook) {
    Serie* res = (Serie*) malloc(sizeof(Serie));
    res->titolo = (char*) malloc(nBook*sizeof(char));
    res->nBook = nBook;
    return res;
}

void setBook(Serie* s, char* name) {
    strcpy(s->titolo, name);
}

int rightWing(Serie* s, int i) {
    return ((s->titolo)[i] > 'd') ? 1 : 0;
}

int main() {
    Serie* s = initSerie(4);
    int i;
    setBook(s,"mad");
    for (i = s->nBook; i > 1; --i) {
        printf("%d\n",rightWing(s,i-2));
    }
    free(s->titolo);
    free(s);
    return 0;
}
```

ESERCIZIO 4 (1 punto)

Il candidato illustri brevemente a che cosa serve il “garbage collector” e le sue funzionalità.

Soluzioni

ESERCIZIO 1

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

list coppie(list list1) {
    if (empty(list1)) {
        return emptyList();
    } else {
        char h= head(list1), hh;
        list temp=tail(list1);
        hh=head(temp);
        if (h<= hh) return cons(h, coppie(tail(temp)));
        else return coppie(tail(temp));

    }
}

int main() {
    list list1 = emptyList();
    list list2;
    list1 = cons('k',list1);
    list1 = cons('o',list1);
    list1 = cons('r',list1);
    list1 = cons('g',list1);
    list2 = coppie(list1);
    while( ! empty(list2) ){
      printf("%c\n", 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 → B → P → (N-B-B) → (1N-B-B) → (10-B-B) → (10-N-B) → (10-1-B) → (10-1-P) → (10-1-(N-B-B)) → (10-1-(1N-B-B)) → (10-1-(11-B-B)) → (10-1-(11-N-B)) → (10-1-(11-0-B)) → (10-1-(11-0-N)) → (10-1-(11-0-1))

ESERCIZIO 3

L'output prodotto è

```
0
0
1
```

La `struct Serie` contiene un puntatore a carattere e un numero.
La funzione `initSerie` riceve in ingresso un numero e restituisce un puntatore a `Serie`. Al suo interno alloca la memoria per la serie, e per un array di caratteri della dimensione del numero in ingresso che è assegnato all'attributo `titolo`, mentre l'attributo `nBook` prende il numero passato in ingresso. La struttura così inizializzata è ritornata in output.
La funzione `setBook` riceve in ingresso un puntatore a `Serie` e una stringa. La stringa viene copiata nell'attributo titolo della serie.
La funzione `rightWing` riceve in ingresso un puntatore a `Serie` e un intero, e restituisce un intero. Esegue un confronto sul carattere alla posizione indicata dall'intero del titolo per decidere se ritornare 1 o 0.
Il `main` inizializza una `struct Serie`, poi chiama la funzione `rightWing` e ne stampa il risultato per ogni carattere dal terzo al primo.