Fondamenti di Informatica T-1 (A.A. 2016/2017) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Giovedì 16 Febbraio 2017 – durata 1h
Totale 12 punti, sufficienza con 7

Compito A

ESERCIZIO 1 (6 punti)

Si scriva una funzione RICORSIVA `filter`

```
list filter(list x, int n)
```

che data una lista `x` d'interi ed un intero `n`, dia in uscita una nuova lista composta dagli elementi di `x` strettamente positivi se `n` è strettamente positivo, dagli elementi di `x` strettamente negativi se `n` è negativo e la lista vuota se `n` è uguale a zero.

Si realizzi una funzione `main()` che crei una lista `x={-3,0,1,4,-6}` ed utilizzi correttamente la funzione `filter(x,n)` per tre volte con `n` che assume i valori `3`, `-2` e `0`, memorizzando i risultati in 3 liste `r1`, `r2` e `r3`. I risultati attesi sono `r1={1,4}`, `r2={-3,-6}` e `r3={}`.

Le funzioni dovranno essere implementate utilizzando le primitive dell'ADT lista, includendo `"list.h"`.

ESERCIZIO 2 (2 punti)

Si consideri la seguente grammatica `G` con scopo `S`, simboli non terminali `{T, D, U}` e simboli terminali `{1, +, -, (, )}`. Si noti che anche le parentesi tonde sono considerate simboli terminali.

```
S := U | D | T
T := (D)+U | U+(D)
D := (T)-U | U+U
U := ((D)-U) | 1
```

La stringa "**(1+(1+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 <stdio.h>
#include <stdlib.h>

typedef struct Win {
    int min,max,badHash;
} TWin;

TWin* create(int min, int max) {
    TWin* res = malloc(sizeof(TWin));
    res->min = min;
    res->max = max;
    res->badHash = min+max;
    return res;
}

int smartEqual(TWin* rA, TWin* rB) {
    if (rA == rB) {
        printf("a");
        return 1;
    }
    if ((rA->badHash) != (rB->badHash)) {
        printf("b");
        return 0;
    }
    printf("c");
    return (rA->min)==(rB->min) && (rA->max)==(rB->max);
}

int main() {
    TWin* r1 = create(1,2);
    TWin* r2 = create(-1,2);
    TWin* r3 = create(1,2);
    TWin* r4 = create(0,3);
    printf("%d\n",smartEqual(r1,r2));
    printf("%d\n",smartEqual(r1,r3));
    printf("%d\n",smartEqual(r1,r4));
    return(0);
}
```

ESERCIZIO 4 (1 punto)

Si scriva un semplice programma C che esegue il confronto fra due stringhe prima leggendole da input e poi stampando la seconda delle due in ordine alfabetico, utilizzando la funzione della libreria C sulle stringhe `strcmp`.

Soluzioni

ESERCIZIO 1

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

list filter(list x, int n) {
    if (empty(x)|| n==0) {
        return emptyList();
    } else {
        int h = head(x);
        if (n > 0 && h > 0) {
            return cons(h, filter(tail(x),n));
        } else if (n < 0 && h < 0) {
            return cons(h, filter(tail(x),n));
        } else {
            return filter(tail(x),n);
        }
    }
}

int main()
{
    list x = emptyList();
    list r1,r2,r3;
    x = cons(-6,x);
    x = cons(4,x);
    x = cons(1,x);
    x = cons(0,x);
    x = cons(-3,x);
    r1 = filter(x,3);
    r2 = filter(x,-2);
    r3 = filter(x,0);
    return 0;
}
```

ESERCIZIO 2

La stringa "**(1+(1+1))-1**" appartiene al linguaggio. In particolare, si può ottenere tramite la seguente derivazione left-most:

S → D → (T)-U → (U+(D))-U → (1+(D))-U → (1+(U+U))-U → (1+(1+1))-U → (1+(1+1))-1

ESERCIZIO 3

L'output prodotto è

```
b0
c1
c0
```

`TWin` è una struttura contenente tre variabili intere.

La funzione `create` riceve due parametri interi. Istanzia con una `malloc` una zona di heap atta a memorizzare il contenuto di una struttura di tipo `TWin`, poi copia i parametri all'interno del `TWin`. Si noti che la malloc non presenta il cast esplicito. Questo non impedisce al programma di compilare correttamente.

Al terzo parametro di `TWin`, `badHash`, è assegnata la somma dei primi due parametri. Il `TWin` così inizializzato è ritornato da `create`.

La funzione `smartEqual` riceve due puntatori a `TWin` e ritorna un `int`. Se i due puntatori sono uguali (puntano alla stessa locazione) `smartEqual` scrive "`a`" e ritorna `1`. Se i campi `badHash` dei due `TWin` puntati sono diversi, `smartEqual` scrive "`b`" e ritorna `0`. Se nessuna di queste due evenienze si verifica, `smartEqual` scrive "`c`" e ritorna `1` se i due `TWin` sono uguali variabile per variabile, `0` altrimenti.

Il `main` dichiara 4 puntatori a `TWin` chiamati `r1`, `r2`, `r3` e `r4`. Ai puntatori assegna il risultato della funzione `create`, variando di volta in volta i parametri della stessa.

Con tre chiamate consecutive di `printf`, il `main` stampa il risultato di tre chiamate di `smartEqual`, dove il primo parametro è sempre `r1`, mentre il secondo parametro va da `r2` a `r4`.

ESERCIZIO 4

```
#include <stdio.h>
#include <stdlib.h>
#define DIM 100

int main() {
    char str1[DIM], str2[DIM];
    scanf("%s", str1);
    scanf("%s", str2);
    if(strcmp(str1, str2) > 0)
        printf("%s", str1);
    else
        printf("%s", str2);
}
```