Fondamenti di Informatica T-1 (A.A. 2015/2016) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Giovedì 7 Luglio 2016 – durata 1h
Totale 12 punti, sufficienza con 7

Compito A

ESERCIZIO 1 (6 punti)

Sia data una lista `l1` di caratteri contenente solo lettere non accentate minuscole e spazi (caratteri bianchi). Si realizzi la funzione RICORSIVA

```
list Maiuscole(list l1);
```

che ritorni una nuova lista `l2` in cui, a partire da `l1`, ogni carattere sia la lettera maiuscola corrispondente alla minuscola contenuta in `l1`, e gli spazi siano rimossi.

Si faccia uso della seguente funzione di conversione di caratteri, contenuta in `ctype.h`:

```
int toupper(int c)
```

Se `c` è una lettera minuscola, `toupper(c)` restituisce la lettera maiuscola corrispondente, altrimenti restituisce `c`.

Si realizzi infine una funzione `main()` che crei una lista `l1` ed utilizzi correttamente la funzione `Maiuscole(l1)` precedente in modo tale da calcolare la lista `l2`. Si definisca
`l1 = {'h','e','l','l','o',' ','w','o','r','l','d'}`
si dovrà ottenere la lista
`l2 = {'H','E','L','L','O','W','O','R','L','D'}`
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 simbolo iniziale `S`, simboli non terminali {`B,N,C`} e simboli terminali {`(,),t,f,n,o,e,v`}

```
S := B | N | (SCS)
B := t | f
N := nS
C := o | e | v
```

La stringa "**n((tof)ent)**" 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>

int bs(int a[], int fi, int ti, int k) {
    int lo, hi, m, mv;
    lo = fi;
    hi = ti - 1;
    while (lo <= hi) {
        m = (lo + hi) / 2;
        mv = a[m];
        if (mv < k) {
            lo = m + 1;
        } else if (mv > k) {
            hi = m - 1;
        } else {
            k=k+1;
            printf("%d\n",k);
            return m;
        }
    }
return -1;
}

int main() {
    int a[] = {1,3,3,5,6,8,11,14};
    int k=6, res = bs(a,1,6,k);
    printf("%d\n",res);
    printf("%d\n",k);
    return(0);
}
```

ESERCIZIO 4 (1 punto)

Si spieghi brevemente cosa sono i linguaggi di alto livello e quali sono i loro punti di forza rispetto ai linguaggi assembler.

Soluzioni

ESERCIZIO 1

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

list Maiuscole(list l) {
    if (empty(l))
       return emptyList();
    else {
        char h = head(l);
        list t = tail(l);
        if (h == ' ')
            return Maiuscole(t);
        else
           return cons(toupper(h),Maiuscole(t));


           }


}


int main() {
    list l1 = emptyList();
    list l2;
    l1 = cons('d',l1);
    l1 = cons('l',l1);
    l1 = cons('r',l1);
    l1 = cons('o',l1);
    l1 = cons('w',l1);
    l1 = cons(' ',l1);
    l1 = cons('o',l1);
    l1 = cons('l',l1);
    l1 = cons('l',l1);
    l1 = cons('e',l1);
    l1 = cons('h',l1);
    l2 = Maiuscole(l1);
    return(0);
}
```

ESERCIZIO 2

La frase appartiene al linguaggio. In particolare, la si può ottenere tramite la seguente derivazione left-most:
S → nS → n(SCS) → n((SCS)CS) → n((BCS)CS) → n((tCS)CS) → n((toS)CS) → n((toB)CS) → n((tof)CS) → n((tof)eS) → n((tof)enS) → n((tof)enB) → n((tof)ent)

ESERCIZIO 3

L'output prodotto è

```
7      4      6
```

Il programma `main` crea un array `a` di interi ordinati, chiama `bs` con parametri `a`, `1`, `6`, `6`, e assegna il risultato all'intero `res`, che è poi stampato.

`bs` usa il primo intero passato, `1`, per inizializzare `lo`, ed il secondo intero passato `ti=6` per inizializzare `hi` a `ti-1`. `k`, pari a `6`, è infine usato come valore di test nei confronti.

`bs` contiene un ciclo while che ripete finchè `lo <= hi`, che al primo passaggio è verificato. A `m` è assegnata la media tra `lo` e `hi`, ossia `3`, mentre a `mv` è assegnato il valore `m`-esimo dell'array `a`, ossia `5`. Il controllo `mv < k`, `5 < 6` in questo caso, è verificato, portando all'esecuzione del ramo then, dove `lo` viene aggiornato ad `m+1`, quindi a `4`.

Al secondo passaggio dalla condizione del while, `lo` vale `4` e `hi` sempre `5`, quindi la condizione è ancora verificata. `m` come risultato di `(lo + hi) / 2` diventa `4` e `mv` diventa `6`. Le condizioni `mv < k` e `mv > k` sono entrambe non verificate, e si entra nell'ultimo ramo else, che ritorna `m`. Nel main esso viene stampato, dunque lo standard output conterrà "`4`". Si noti invece che k è passato per valore e quindi l'incremento viene attualizzato solo all'interno della funzione.

Si può notare che `bs` implementa la ricerca binaria su un sottointervallo di un array ordinato di interi. I parametri indicano l'inizio e la fine del sottointervallo, oltre ovviamente al valore da cercare. Il valore ritornato è la posizione dell'elemento cercato nell'array, oppure `-1` se l'elemento cercato non è presente nel sottointervallo specificato. Nella chiamata fatta nel main, il valore cercato `6` è infatti alla posizione `4`.