Fondamenti di Informatica T-1 (A.A. 2018/2019) - Ingegneria Informatica

Prof.ssa Mello

Prova Scritta – 10 Gennaio 2019 – durata 1h

Totale 12 punti, sufficienza con 7

Compito B

ESERCIZIO 1 (6 punti)

Si scriva una funzione **ITERATIVA** `generateList`

```
list generateList(int n, int m)
```

che, dati due numeri interi `n` e `m`, dia in uscita una lista composta dagli interi compresi fra `n` e `m` (estremi inclusi) in senso crescente se `n < m`, decrescente se `n > m`, mentre se `n = m` verrà restituito nella lista solo il valore `n`.

Si realizzi una funzione `main()` che utilizzi correttamente la funzione `generateList(int n, int m)` chiamandola due volte con `n` che assume il valore `6`, e `m` il valore `8`, e poi, viceversa con `n` che assume il valore `8`, e `m` il valore `6`, e memorizzando i risultati in due liste `l1` e `l2` rispettivamente. Il risultato atteso è `l1=[6,7,8] e l2=[8,7,6].` La lista risultante `l1` deve poi essere stampata a terminale.

La funzione dovrà essere implementata utilizzando le primitive dell'ADT lista, includendo `"list.h"`.

ESERCIZIO 2 (2 punti)

Un elaboratore rappresenta i numeri interi su 8 bit tramite la notazione in complemento a 2. Indicare come viene svolta la seguente operazione aritmetica calcolandone il risultato secondo la rappresentazione binaria in complemento a 2 (si trasli anche il risultato in decimale per verificare la correttezza dell'operazione):
10-20.

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 check(int* n0, int n1, char c) {
    switch (c) {
    case 'M':
        return (*n0) > n1 ? 0 : 1;
    case 'm':
        return (*n0) < n1 ? 0 : 1;
    }
}

int checkEach(int* p, int l, char c) {
    int* ind = p;
    int i;
    int count = 0;
    for (i = 0; i < l-1; i++) {
        count += check(ind, ind[1], c);
        printf("%d\n", count);
        ind++;
    }
    return 0;
}

int main(){
    int a[] = {3,-4,-1,2};
    checkEach(a,4,'m');
    return 0;
}
```

ESERCIZIO 4 (1 punto)

Si descriva brevemente come può essere espressa formalmente la grammatica (sintassi) per un linguaggio e se ne mostri un semplice esempio.

Soluzioni

ESERCIZIO 1

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

list generateList(int n, int m) {
      int i;
      list l = emptyList();
      if (n == m)
         return cons(n, l);
      else{
            if (n < m){
                  for(i=m; n<=i; i--)
                        l = cons(i,l);
            }
            else{
                  for(i=m; i<=n; i++)
                        l = cons(i,l);

            }
      }
      return l;
}

int main()
{
      list l1,l2;
            l1 = generateList(6,8);
            l2 = generateList(8,6);
            while(!empty(l1) ){
                  printf("%d\n", head(l1));
                  l1 = tail(l1); }


      return 0;
}
```

ESERCIZIO 2

10 - 20 = - 10
10 → **00001010**
20 → 00010100
- 20 → **11101100**
10 – 20 → **11110110**
- 11110110 = 00001010 → 10

ESERCIZIO 3

Il programma è corretto e l'output prodotto è:

```
1
1
1
```

La funzione `main` inizializza un array `a` e invoca la funzione `checkEach` passando come parametri `a`, l'intero `4` e il carattere '`m`'.

La funzione `checkEach` riceve in input un puntatore a intero `p`, un intero `l` e un carattere `c`. `p` diventa quindi puntatore all'array `a`, mentre `l` assume il valore `4` e `c` il valore '`m`'.
Viene creato un nuovo puntatore a intero `ind` che viene inizializzato come puntatore ad `a`, viene dichiarato un intero `i` e un altro intero `count` viene inizializzato a `0`. All'inizio del ciclo `for`, `i` assume il valore `0`; il ciclo si ripeterà finché `i` sarà minore di `l-1`, ovvero `4-1`, ovvero `3`.
Poiché `0 < 3`, si entra nel ciclo: all'intero `count` è sommato il risultato della funzione `check`. La funzione `check` è invocata passando come parametri il puntatore `ind`, il valore contenuto nella cella di memoria successiva a quella riferita dal puntatore `ind` e il carattere `c`. Quindi in questo caso viene invocata passando come parametri il puntatore alla prima cella dell'array `a`, il valore `-4` e '`m`'.

All'interno della funzione `check`, poiché il carattere `c` ha valore '`m`', si entra nel secondo caso. Il programma testa se il valore contenuto all'indirizzo `n0` è minore del valore `n1`, ovvero se `3` è minore di `-4`. Poiché ciò non è vero, viene restituito il valore `1`.

All'interno della funzione `checkEach`, la variabile `count` viene quindi incrementata a 1, che viene stampato. Successivamente il valore del puntatore `ind` è incrementato, ciò significa che punterà alla seconda cella dell'array `a`. L'intero `i` è incrementato di `1`, assumendo il valore `1`. Poiché `1 < 3`, il ciclo prosegue.
La funzione `check` è invocata passando l'indirizzo alla seconda cella di `a` e il valore `-1`. Poiché il contenuto della cella, `-4`, è minore di `-1`, viene restituito dalla funzione il valore `0`. Il valore di `count` rimane quindi a `1`, che viene stampato. Il valore di `ind` viene incrementato, ora punta alla terza cella di `a`. Il valore di `i` viene incrementato a `2`. Poiché `2 < 3`, il ciclo prosegue.
La funzione `check` è invocata passando l'indirizzo alla terza cella di `a` e il valore `2`. Poiché il contenuto della cella, `-1`, è minore di `2`, viene restituito dalla funzione il valore `0`. Il valore di `count` rimane quindi `1`, che viene stampato. Il valore di `ind` viene incrementato, ora punta alla quarta cella di `a`. Il valore di `i` viene incrementato a `3`; poiché `3 < 3` non è vero, il ciclo termina.
Dopodiché il programma termina.