Fondamenti di Informatica T-1 (A.A. 2017/2018) - Ingegneria Informatica
Prof.ssa Mello
Prova Scritta - 11 Gennaio 2018 – durata 1h
Totale 12 punti, sufficienza con 7

Compito A

ESERCIZIO 1 (6 punti)
Data una lista di interi `list1` e un intero `div`, si realizzi una funzione ITERATIVA

```
list separate(list list1, int div);
```

che restituisca una nuova lista contenente gli elementi di `list1` che sono divisibili in modo esatto per `div`.

Ad esempio se `list1 = [9, 2, 6, 7, 3]` e `div = 3`, la funzione `separate()` deve restituire la lista `[9,6,3]`, ovvero i soli valori della lista `list1` che rispettano il requisito dato.

Si realizzi inoltre una funzione RICORSIVA che restituisca l'ultimo elemento di una lista (supposta avere almeno un elemento).

```
int last(list list1);
```

Ad esempio se `list1 = [9, 2, 6, 7, 3]`, la funzione `last()` deve restituire il valore `3`.

Le funzioni dovranno essere implementate 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 `separate()` istanziando `div` all'ultimo elemento della delle lista `list1`.

Nota: l'ordine degli elementi della lista restituita dalla funzione `separate()` è ininfluente, nell'esempio va bene anche se restituisce la lista `[3,6,9]`

ESERCIZIO 2 (2 punti)

Si consideri la seguente funzione

```
float attiva(float c, int i){
      if( i <= 1 ){
            return 1;
      }
      else{
            float j = c - 1;
            return j + attiva(j, i-1);
      }
}
```

Mostrare la sequenza dei record di attivazione ed il valore di ritorno nel caso in cui la funzione sia invocata con parametri attuali (14.5, 4).

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 R(char* fvec, int P, int N){
  int i;
  for (i=4; i > (i?P:N); i--){
    if (i==1)
      fvec[i-1]='a';
    printf("%c", fvec[i-1]);
    N--;
  }
  return N;
}

int main(){
 int P=0;
 int N=12;
 char s[]= "osso";
 int r=R(s,P,N);
 printf("\nN=%d r=%d\n",N,r);
 return 0;
}
```

ESERCIZIO 4 (1 punto)

Il candidato illustri brevemente il significato e l'utilizzo delle variabili esterne (introdotte dalla parola chiave EXTERN).

Soluzioni

ESERCIZIO 1

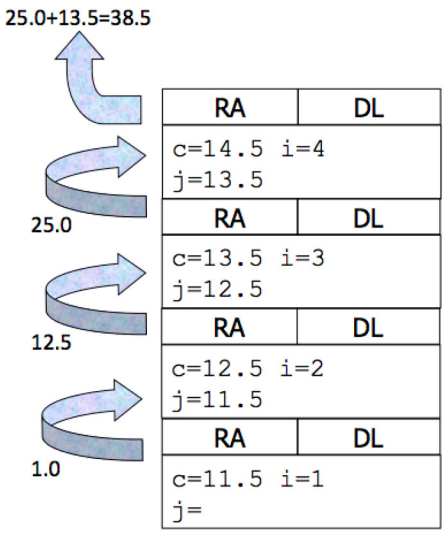
```
#include <stdio.h>
#include "list.h"

// ITERATIVA
list separate (list l, int n){
  list res = emptylist();
  while( !empty(l)){
    if( (head(l)%n)==0){
      res = cons(head(l), res);
    }
    l = tail(l);
  }
  return res;
}

// RICORSIVA
int last (list l){
  if (empty(tail(l)))  return head(l);
  else  return last(tail(l));
}

int main(){
  list l, res;
  l = cons(9, cons(2, cons(6, cons(7, cons(3, emptylist())))));
  res = separate(l,last(l));
  while( ! empty(res) ){
    printf("%d\n", head(res));
    res = tail(res);
  }
  return 0;
}
```

ESERCIZIO 2



ESERCIZIO 3

L'output prodotto è

```
ossa
N=12 r=8
```

Il `main` inizializza `P=0 N=12` e crea una stringa `s`, contenente `s="osso\0"` e poi invoca la funzione `R` con tali parametri. La variabile `s` viene passata per riferimento, `N` e `P` per valore.

All'interno della funzione, viene dichiarata la variabile `i` che assume valore `5` all'esecuzione del ciclo `for`.

La condizione del `for` prevede l'esecuzione di due passi successivi: prima viene testato il suo valore di `i`. Se tale valore è diverso da `0`, l'espressine tra parentesi viene sostituita con `P` altrimenti con `N`. Poi viene testata la condizione `>`.

Alla prima esecuzione del `for` si ha `i=4`. Poiché `4!=0`, si testa la condizione `i>P`, ovvero `4>0`. All'interno del `for`, poiché `i==1` non è soddisfatta, si eseguono solo la `printf` (che stampa il carattere `fvec[4-1]='o'`) e il post decremento di `N` (che diventa `11`).

Alla seconda esecuzione del `for` si ha `i=3`. Poiché `3!=0`, si testa la condizione `i>P`, ovvero `3>0`. All'interno del `for`, poiché `i==1` non è soddisfatta, si eseguono solo la `printf` (che stampa il carattere `fvec[3-1]='s'`) e il post decremento di `N` (che diventa `10`).

Alla terza esecuzione del `for` si ha `i=2`. Poiché `2!=0`, si testa la condizione `i>P`, ovvero `2>0`. All'interno del `for`, poiché `i==1` non è soddisfatta, si eseguono solo la `printf` (che stampa il carattere `fvec[2-1]='s'`) e il post decremento di `N` (che diventa `9`).

Alla quarta esecuzione del `for` si ha `i=1`. Poiché `1!=0`, si testa la condizione `i>P`, ovvero `1>0`. All'interno del `for`, poiché `i==1` è ora soddisfatta, si esegue `fvec[1-1]='a'` (quindi la stringa `s` del `main` diventa `"asso\0"`). Infine si eseguono la `printf` (che stampa il carattere `fvec[1-1]='a'`) e il post decremento di `N` (che diventa `8`).

Alla quinta esecuzione del `for` si ha `i=0`. Poiché `0!=0` non è soddisfatta, si testa la condizione `i>N`, ovvero `0>8`. La condizione non è soddisfatta e il ciclo termina.

La funzione ritorna `N=8`, che il `main` salva nella variabile `r`. Poiché `N` è stata passata per copia, il suo valore nel `main` non è modificato e l'ultima stampa risulta quindi: `N=12 r=8`.