**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 B

**ESERCIZIO 1** (6 punti)
Data una lista di interi `intlist` e un intero `b`, si realizzi una funzione RICORSIVA

```
list find_div(list intlist, int b);
```

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

Ad esempio se `intlist = [9, 6, 5, 4, 2]` e `b = 2`, la funzione `find_div ()` deve restituire la lista `[6,4,2]`, ovvero i soli valori della lista `intlist` che rispettano il requisito dato.

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

```
int ultimo(list intlist);
```

Ad esempio se `intlist = [9, 6, 5, 4, 2]`, la funzione `last()` deve restituire il valore `2`.

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 `find_div ()`istanziando `b` all'ultimo elemento della delle lista `intlist`.

Nota: l'ordine degli elementi della lista restituita dalla funzione `find_div()` è ininfluente, nell'esempio va bene anche se restituisce la lista `[2,4,6]`

**ESERCIZIO 2** (2 punti)

Si consideri la seguente funzione

```
float record(float v, int x){
      if( x <= 1 ){
            return 1;
      }
      else{
            float j = v - 1;
            return j + record(j, x-1);
      }
}
```

Mostrare la sequenza dei record di attivazione ed il valore di ritorno nel caso in cui la funzione sia invocata con parametri attuali (13.5, 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 Z(char* fvec, int Q, int M){
  int j;
  for (j=4; j > (j?Q:M); j--){
    if (j==1)
      fvec[j-1]='o';
    printf("%c", fvec[j-1]);
    M--;
  }
  return M;
}

int main(){
 int Q=0;
 int M=12;
 char s[]= "inni";
 int r=Z(s,Q,M);
 printf("\nM=%d r=%d\n",M,r);
 return 0;
}
```

**ESERCIZIO 4** (1 punto)

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

# Soluzioni

## ESERCIZIO 1

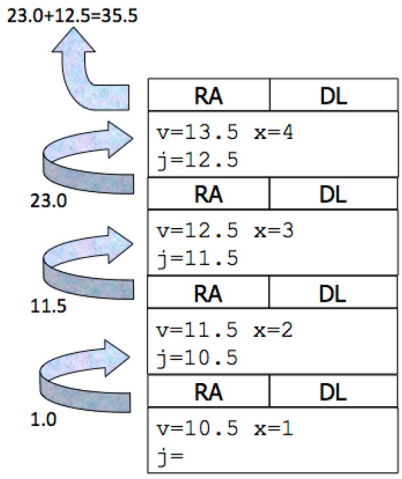
```
#include <stdio.h>
#include "list.h"

// RICORSIVA
list find_div (list l, int n){
  if (empty(l))
    return emptylist();
  else
    if( (head(l)%n)==0){
      return cons(head(l), find_div(tail(l), n));
    }else{
      return find_div(tail(l), n);
    }
}

// ITERATIVA
int ultimo(list l){
list res = emptylist();
  while( !empty(tail(l)))
    l = tail(l);
  return head(l);
  }

int main(){
  list l, res;
  l = cons(9, cons(6, cons(5, cons(4, cons(2, emptylist())))));
  res = find_div(l,ultimo(l));
  while( ! empty(res) ){
    printf("%d\n", head(res));
    res = tail(res);
  }
  return 0;
}
```

## ESERCIZIO 2



**ESERCIZIO 3**

L'output prodotto è

```
inno
M=12 r=8
```

Il `main` inizializza `Q=0 M=12` e crea una stringa `s`, contenente `s="inni\0"` e poi invoca la funzione `Z` con tali parametri. La variabile `s` viene passata per riferimento, `M` e `Q` per valore.

All'interno della funzione, viene dichiarata la variabile `j` 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 `j`. Se tale valore è diverso da `0`, l'espressine tra parentesi viene sostituita con `Q` altrimenti con `M`. Poi viene testata la condizione `>`.

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

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

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

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

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

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