FOND. DI INFORM. & PROGR. A OGGETTI **SCRITTO 12 GENNAIO 2012**
CORSO DI LAUREA IN INGEGNERIA INFORMATICA

NOTE SULLO SVOLGIMENTO DELLA PROVA SCRITTA:

- SCRIVERE IL PROPRIO NOME, COGNOME E NUMERO DI MATRICOLA SU OGNI FOGLIO UTILIZZATO PER LO SVOLGIMENTO DELLA PROVA
- RICONSEGNARE TUTTI I FOGLI. NON SCRIVERE A MATITA.
- SPEGNERE I TELEFONINI
- NON È POSSIBILE UTILIZZARE CALCOLATRICI
- È POSSIBILE CONSULTARE SOLO LA DISPENSA SUL LINGUAGGIO ASSEMBLER DISPONIBILE SULLA CATTEDRA
- I PRIMI DUE ESERCIZI VALGONO 10 PUNTI; GLI ULTIMI 2 VALGONO 5 PUNTI
- L'ESERCIZIO 4 È DIVERSO A SECONDA DELL'A.A. IN CUI È STATO SEGUITO IL CORSO
- TEMPO PER LA PROVA 2 ORE

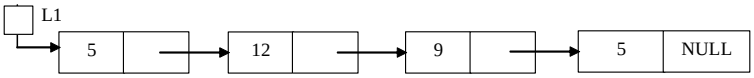
ESERCIZIO 1

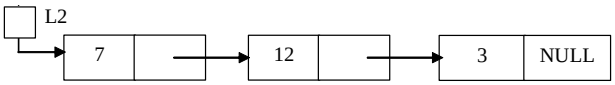
Sia data la struttura seguente:

```
struct elem {int info; elem* pun;};
```

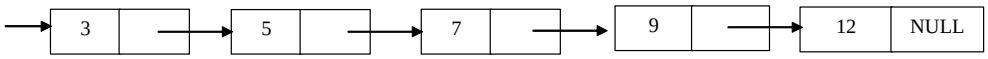
Scrivere una funzione che, date due liste `L1` ed `L2` di elementi di tipo `elem` passate come argomento alla funzione, restituisce una nuova lista che contiene un elemento per ogni elemento di `L1` ed `L2` senza duplicati. La lista restituita deve inoltre essere ordinata per valori crescenti del campo informazione.

Per esempio, se la funzione viene chiamata con le liste L1 ed L2 seguenti,





la funzione restituisce la lista:



ESERCIZIO 2

Scrivere una funzione che prende in ingresso un numero intero `n>0` ed un file, e restituisce una matrice quadrata di interi di dimensione `nxn` inizializzata come segue. Se l'intero `n` è pari, gli elementi della matrice sono tutti uguali a 0 eccetto quelli sulle diagonali (principale e secondaria), che devono assumere valore 1. Se l'intero `n` è dispari, gli elementi della matrice sono letti dal file. La matrice è inizializzata per colonne. I numeri interi letti dal file corrispondono al valore degli elementi della prima colonna, seguiti da quelli della seconda colonna e cosi' via. Se non ci sono interi nel file sufficienti a coprire l'inizializzazione di tutti gli elementi della matrice, gli elementi mancanti devono essere inizializzati a 0.

Per esempio, se la funzione viene chiamata con numero intero `n` uguale a `5` e con un file il cui il contenuto è il seguente:

```
3   1   3   5  9   5   7 eof
```

la matrice restituita è (dimensione `5x5`):

```
3  5  0  0  0
1  7  0  0  0
3  0  0  0  0
5  0  0  0  0
9  0  0  0  0
```

ESERCIZIO 3

Sia data la struttura seguente:

```
struct elem {char info; elem* pun;};
```

Scrivere una funzione ricorsiva che, data una lista di elementi di tipo `elem` ed un carattere `c`, restituisce `true` se gli elementi in posizione pari hanno campo informazione uguale a `c`. Altrimenti la funzione restituisce `false`. Assumere la posizione degli elementi indicata a partire da 1. Il primo elemento della lista ha posizione 1.

ESERCIZIO 4 (Anno accademico 2011-2012)

1) Rappresentare l'intero (-75) in complemento a 2 su 8 bit.

2) Indicare il contenuto del registro AL al termine del programma Assembler seguente, nel caso in cui venga letto da tastiera il numero naturale 3 (attraverso due cifre esadecimali `0 3`):

```
#.GLOBAL _main
 .EQU V,90

# Sezione codice
_main: CALL inbyte
        CALL newline
        MOV AL,CL
        MOV $V,AL
        MOV $V,BL

loop:  ADD AL,BL
       JC fine
       DEC CL
       JNZ loop

fine:  MOV CL,AL
     # Emissione del risultato dell'elaborazione
     CALL outbyte
     CALL pause
     RET
```

NOTE

Nome: **inbyte**

Azione: Attende che dalla tastiera arrivino 2 (codifiche ASCII di) cifre esadecimali. Quando ciò avviene deposita nel registro AL il byte espresso in forma compatta dalle due cifre. Oltre a ciò fa l'eco sul monitor (delle codifiche ASCII) delle 2 cifre esadecimali prelevate

Nome: **outbyte**

Azione: Visualizza sul monitor il contenuto del registro AL sottoforma di due cifre esadecimali.

ESERCIZIO 4 (Anni accademici precedenti al 2011-2012)

1) Rappresentare l'intero (-75) in complemento a 2 su 8 bit.

2) Si mostri l'uscita a video del programma C++ seguente:

```
#include <iostream>
using namespace std;

class A{
public:
     int x;
     A(int n=0) {  x=n; cout << "A:x="<< x << endl;}
     void f() { cout << "A:f() x=" <<  x << endl;}
     virtual ~A() {  cout << "via A"  << endl; }
};

class B: public A{
A uno;
public:
      B(int n=7):uno(n) { x=n;  cout<<"B:x="<<x<< endl; }
      virtual void f() { cout << "B:f() x=" <<  x << endl; }
     ~B() {  cout << "via B"  << endl; }
} ;

class C: public B{
    B due;
    int x;
public:
      C(int n=1) :B(n) { x=n;  cout<<"C:x="<< x << endl; }
      void f() { cout << "C:f() x=" <<  x << endl; }
      ~C() {  cout << "via C"  << endl; }
};

int main(){
   C* pc = new C(3);
   B* pb = pc;
   A* pa = pc;
   pa->f();
   pb->f();
   pc->f();
   delete pb;
   return 0;
}
```