FONDAMENTI DI INFORM. & PROGR. A OGGETTI **SCRITTO 8 FEBBRAIO 2012**
CORSO DI LAUREA IN INGEGNERIA INFORMATICA

NOTE SULLO SVOLGIMENTO DELLA PROVA SCRITTA:

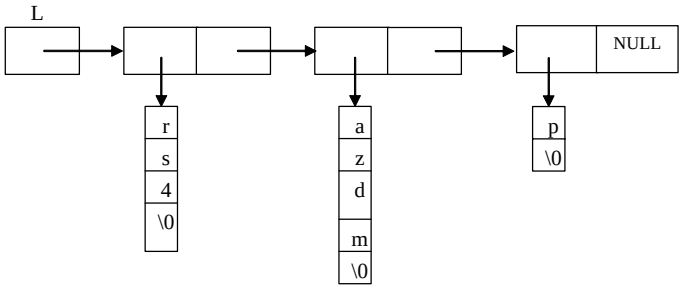
- SCRIVERE IL PROPRIO NOME, COGNOME E NUMERO DI MATRICOLA SU OGNI FOGLIO UTILIZZATO PER LO SVOLGIMENTO DELLA PROVA
- RICONSEGNARE TUTTI I FOGLI. NON SCRIVERE A MATITA.
- SPEGNERE I TELEFONINI
- NON È POSSIBILE UTILIZZARE CALCOLATRICI
- È POSSIBILE CONSULTARE SOLO LA DISPENSA SUL LINGUAGGIO ASSEMBLER 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

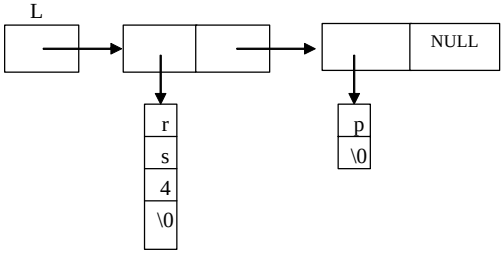
Sia data la struttura seguente, con il campo `info` puntatore ad una stringa allocata nella memoria dinamica:

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

Scrivere una funzione che data una lista `L` di elementi di tipo `elem` ed una stringa `st`, elimina dalla lista `L` tutti gli elementi il cui campo informazione è minore o uguale a `st`. La lista e la stringa devono essere passate come argomento alla funzione. Per esempio, se la funzione viene chiamata con la lista `L` seguente e la stringa `"ciao"`,



la lista `L` viene modificata come segue:



ESERCIZIO 2

Sia dato il tipo seguente:

```
enum forma {X, O, T};
```

Scrivere una funzione che prende in ingresso una matrice quadrata di dimensione `nxn` di elementi di tipo `forma`, ed un intero `K`. La funzione restituisce `true` se esistono almeno `K` elementi consecutivi della stessa forma su una riga oppure su una colonna della matrice; la funzione restituisce `false` altrimenti. Nello scrivere la funzione si supponga che K sia sempre maggiore di 0.

Per esempio, se la funzione viene chiamata con la matrice 5x5 seguente e con l'intero 3, la funzione restituisce `false`.

Se la funzione viene chiamata con la matrice 5x5 seguente e con l'intero 2, la funzione restituisce `true.`

```
X  T  X  T  T
T  O  O  T  O
O  T  X  O  X
T  T  O  X  X
O  O  X  O  O
```

ESERCIZIO 3
Scrivere una funzione ricorsiva che dato un numero naturale N dispari, stampi su uscita standard un maggiore (>) composto di N asterischi. Nello scrivere la funzione si supponga che N sia sempre dispari.
Per esempio, dato N = 5, la funzione stampa su uscita standard

```
  *
   *
    *
   *
  *
```

ESERCIZIO 4 (Anno accademico 2011-2012)
1) Data la rappresentazione $(A0D)_{16}$ in base 16, trasformarla in base 8.

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,40

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

loop:  SHR  $1, BL
       DEC CL
       JNZ loop
       JMP fine

fine: MOV BL, 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) Data la rappresentazione $(A0D)_{16}$ in base 16, trasformarla in base 8.

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;}
     virtual void f() { cout << "A:f() x=" <<  x << endl;}
     void g() { cout << "A:g() x=" <<  x << endl;}
     ~A() {  cout << "via A"  << endl; }
};

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

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

int main(){
  {  C vettore[2];
     A* pa = &vettore[0];
     B* pb = &vettore[1];
     pa-> f();
     pa->g();
     pb-> f();
     pb->g();
   }
   return 0;
}
```