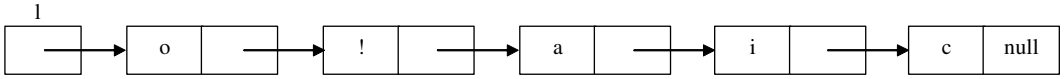
NOTE SULLO SVOLGIMENTO DELLA PROVA SCRITTA:

- **SPEGNERE I TELEFONINI;**
- **SCRIVERE IL PROPRIO NOME, COGNOME E NUMERO DI MATRICOLA SU OGNI FOGLIO UTILIZZATO;**
- **NON È POSSIBILE CONSULTARE NESSUN TIPO DI MATERIALE;**
- **NON È POSSIBILE UTILIZZARE CALCOLATRICI;**
- **PRIMA DI SCRIVERE LA SOLUZIONE DELL'ESERCIZIO, INSERIRE IL NUMERO DI ESERCIZIO CHE SI STA RISOLVENDO. PER ESEMPIO, SCRIVERE "ESERCIZIO N. 1" QUANDO SI STA RISOLVENDO L'ESERCIZIO N. 1;**
- NON COPIARE DAL VICINO (NON È DETTO CHE IL VICINO SIA PIÙ BRAVO DI VOI);
- NON PERMETTETE AL VICINO DI COPIARE (È SPIACEVOLE VEDERSI ANNULLARE IL COMPITO SENZA COLPE);
- TUTTI GLI ESERCIZI VALGONO 5 PUNTI
- **ALLA FINE DELLA PROVA, RICONSEGNARE TUTTI I FOGLI UTILIZZATI.**

1) Sia data la struttura seguente

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

Scrivere una funzione ricorsiva che data una lista di elementi di tipo elem stampa su uscita standard i campi c corrispondenti a lettere dell'alfabeto (si considerino solo lettere minuscole) nell'ordine inverso rispetto ad una visita effettuata dalla testa. Sia data, per esempio, la lista seguente

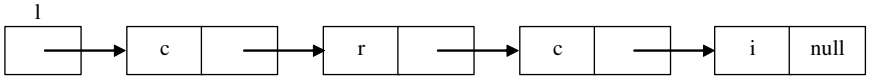


La funzione stamperà su uscita standard "ciao".

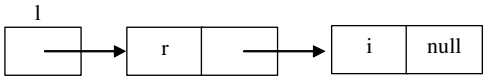
2) Sia data la struttura seguente

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

Scrivere una funzione che, data una lista di elementi di tipo elem ed un carattere, elimina dalla lista tutti gli elementi che hanno il campo x uguale al carattere dato.
Sia data, per esempio, la lista seguente ed il carattere 'c'



Dopo la chiamata alla funzione, la lista viene modificata come segue:



3) Scrivere una funzione che, dato un vettore di stringhe, salvi le stringhe in un file di nome "backup" dopo averle invertite. Le stringhe sono salvate su righe diverse e nello stesso ordine in cui appaiono nel vettore. Si supponga che le stringhe abbiano dimensione costante N=10 e che il vettore possa avere dimensione qualunque.
Per esempio, il vettore [paolo, piero, pino] viene salvato nel file nel modo seguente:

```
oloap
oreip
onip
```

4) Data la rappresentazione $(AE34)_{16}$ in base 16, trasformarla in base 2.
Data la rappresentazione in complemento a due $(11011110)_2$, esprimere il numero in base 10.

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

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

class A {
public:
     int x;
     A () {  x=10; cout << "A:x="<< x << endl;}
     void stampa()
        { x++; cout << "A:stampa() a=" <<  x << endl;}
};

class B: public A {
public:
      int x;
      B (int b = 1) {  x=b; cout << "B:x="<< x << endl; }
      virtual void stampa()
       { cout << "B:stampa() x=" <<  x << " A::x=" << A::x << endl; }
} ;

class C: public B{
     A a;
public:
      C(int c) : B(c) {x=c; cout << "C:x=" << x << endl; }
      void stampa()
        { cout << "C:stampa() x=" <<  x
                 << " A::x=" << A::x << "   a.x=" << a.x << endl; }
};
```

```
int main(){
   C* pc = new C(20);
   A* pa=pc;
   B* pb=pc;
   pa->stampa();
   pb->stampa();
   pc->stampa();
   pb =new B(10);
   pb->stampa();
   return 0;
}
```

6) Si mostri l'uscita a video dell'esecuzione del programma C++ seguente.

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

template<class R, class T>
class CC {
  public:
        CC(R r) { cout << r << endl;}
        R f(R r,T t) { return r+t;}
};

template<class A, class B>
A f (A r, B s, B t) {
    static char c = 'a';
    CC <A, B> cc(r);
    c++;
    cout << "c = " << c << endl;
    return cc.f(r,s+t);
}
```

```
int main () {
   cout << f(2.4, 2, 2) << endl;
   cout << f<double,int>(1, 4, 1) << endl;
   cout << f<int,int>(4, 4, 1) << endl;
   return 0;
}
```