## NOTE SULLO SVOLGIMENTO DELLA PROVA SCRITTA:

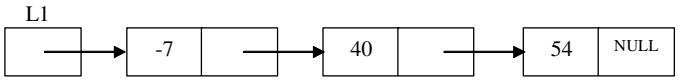
- **SPEGNERE I TELEFONINI;**
- **SCRIVERE IL PROPRIO NOME, COGNOME E NUMERO DI MATRICOLA SU OGNI FOGLIO UTILIZZATO;**
- **NON È POSSIBILE CONSULTARE NESSUN TIPO DI MATERIALE;**
- **NON È POSSIBILE UTILIZZARE CALCOLATRICI;**
- **PRIMA DI SCRIVERE LA SOLUZIONE DELL'ESERCIZIO, INSERIRE IL NUMERO DI ESERCIZIO 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);
- I PRIMI TRE ESERCIZI VALGONO 6 PUNTI; GLI ULTIMI 3 VALGONO 4 PUNTI.
- **ALLA FINE DELLA PROVA, RICONSEGNARE TUTTI I FOGLI UTILIZZATI.**

1) Scrivere una funzione ricorsiva che, dato un vettore di lunghezza qualsiasi composto di elementi di tipo intero, aggiunge 10 a tutti gli elementi minori di 100, e lascia gli altri inalterati. Per esempio, se il vettore è [2, 200, 120, -3], il vettore viene modificato in [12, 200, 120, 7].

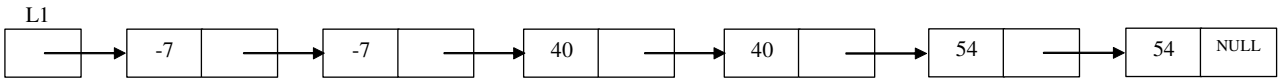
2) Sia data la struttura seguente

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

Scrivere una funzione che, data una lista di strutture di tipo `elem`, modifica la lista creando una replica di ogni elemento. L'elemento replicato deve essere adiacente all'elemento stesso. Per esempio, data la lista



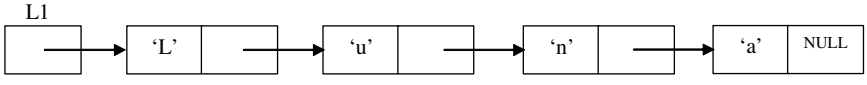

la lista viene modificata come segue:




3) Sia data la struttura seguente

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

Scrivere una funzione che, data una stringa, restituisce una lista di strutture di tipo `elem`, dove il campo `lettera` di ciascuna struttura contiene una lettera della stringa. L'elemento i-esimo della lista deve contenere la lettera i-esima della stringa. Per esempio, data la stringa "Luna", la funzione restituisce la seguente lista:




4) Data la rappresentazione $(FCBA)_{16}$ in base 16, trasformarla in base 2.
Data la rappresentazione $(11000101)_{compl2}$ in complemento a due su 8 bit, trasformarla in base 10.

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

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

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

class B: public A{
public:
      B() { x++;  cout << "nuovo B"<< endl; }
      virtual void f() { cout << "B::f() x=" << x <<endl; }
} ;

class C: public B{
public:
      C() { cout << "nuovo C"<< endl; }
      void f() { x+=10; cout << "C::f() x=" << x << endl; }
};
```

```
class D:  public A{
     B obj;
public:
     D(int n=0): A(n) { x+=n; cout << "nuovo D"<< endl; }
     void f() { cout << "D::f() x=" << x << endl; }
};

int main(){
   C objC;
   B* pb = &objC;
   A* pa = pb;
   pa->f();
   pb->f();

   D objD(7);
   objD.f();
   return 0;
}
```

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

```
#include<iostream>
using namespace std;
template<class R, class S>
class A {
  public:
            void f (R i, S d) {
                     cout << "i = " << i << endl;
                     cout << "d = " << d << endl;
           }
};

template<class R, class S, class T>
T f (R r, S s, T t) {
   static int x = 8;
   A<R, S> a;
   a.f (r, 7);
   x++;
   cout << "x = " << x << endl;
   return t;
}
```

```
int main () {
        cout << f(2.4, 2, 1.1) << endl;
        cout << f<double,double,int>(1.0, 4, 1) << endl;
        cout << f<double,int>(4, 4, 1.1) << endl;
         return 0;
   }
```