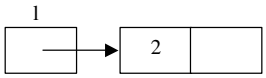
NOTE SULLO SVOLGIMENTO DELLA PROVA SCRITTA:

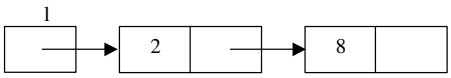
- **SPEGNERE I TELEFONINI;**
- **SCRIVERE IL PROPRIO NOME, COGNOME E NUMERO DI MATRICOLA SU OGNI FOGLIO UTILIZZATO;**
- **NON È POSSIBILE CONSULTARE NESSUN TIPO DI MATERIALE;**
- **NON È POSSIBILE UTILIZZARE CALCOLATRICI;**
- **PRIMA DI SCRIVERE LA SOLUZIONE DELL'ESERCIZIO, INSERIRE IL NUMERO DI ESERCIZIO 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) Sia data la struttura seguente

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

Scrivere una funzione ricorsiva che, data una lista l di strutture di tipo elem ed un intero x, inserisce in fondo alla lista una nuova struttura con campo info uguale ad x.
Per esempio, data la lista l seguente e l'intero x = 8, la funzione modifica la lista come indicato di seguito

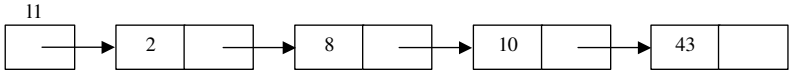


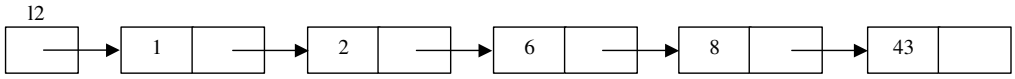


2) Sia data la struttura seguente

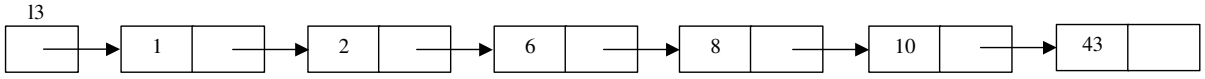
```
struct elem{int info; elem* pun;};
```

Scrivere una funzione che, date due liste l1 e l2 ordinate per **valori crescenti e non duplicati** del campo info, crea una terza lista l3 ordinata per **valori crescenti e non duplicati** del campo info che contiene l'unione delle liste precedenti. Se tra l1 e l2 esistono elementi con lo stesso campo info, solo un elemento deve essere presente in l3. Per esempio, date le due liste l1 e l2





la funzione restituisce la lista l3 seguente:



3) Scrivere una funzione che, data una matrice m(i,j) di interi, con i e j gli indici di riga e di colonna, restituisca la somma degli elementi in posizione (i,j), in cui almeno uno tra l'indice di riga i e l'indice di colonna j siano multipli di 2. La matrice m(i,j) ha dimensione NxM, dove N e M sono variabili. Per esempio, data la matrice seguente

1 2 3
4 5 6
7 8 9
2 3 0

la funzione restituisce 33.

4) Data la rappresentazione $(1EAA29)_{16}$ in base 16, trasformarla in base 2.
Data la rappresentazione in complemento a due $(10100101)_{compl2}$, esprimere il numero in base 10.

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

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

class A {
protected:    int n;
public:
    A(int x) { n = x; n++ ;    cout << "A = " << n << endl;}
    void f1() {cout << 'A' << endl;}
} ;

class B {
   int n;
public:
   B() { n = 10; cout << "B = " << n << endl;}
};

class C: public A {
   B b;
public:
   C() : A(3) {n++; cout << "C = " << n << endl;}
   C(int x) : A(x) {n=x; cout << "C = " << n << endl;}
   virtual void f1(){ cout << 'C' << endl;}
};
```

```
class E {
      B b;
      C c;
public:
       E():c(5) {cout << 'E' << endl;}
};

int main(){
    C* pc = new C;
    A* pa = pc;
    pc->f1();
    pa->f1();
    E e;
    return 0;
}
```

6) Si mostri l'uscita a video delle esecuzioni del programma C++ seguente con rispettivamente gli ingressi c = 'a', c = 'b' e c = 'c'.

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

class E{
 public:
    char c;
    E() {c = '?'; cout << c << endl;}
    virtual void f(){cout << "Ecc " << c << endl;}
};

class E1: public E {
 public:
    E1() { c = '1'; }
};

class E2: public E {
 public:
    E2() { c = '2'; }
};

void f1(char c) {
   try {
   if( c=='a' )
     throw E1();
   }
   catch(E& e) {
      e.f();
    }
}
```

```
char f(char c) {
   try
   {
       f1(c);
       if (c == 'a')
          throw E1();
       else
          if (c == 'b')
             throw E2();
          else throw E();
   }
   catch(E1& e) {
     e.f();
    }
    return c;
}

int main() {
    char c;
    cin >> c;
    try  { cout << f(c) << endl; }
    catch (E& e) {cout << "Ecc. Main" << endl;}
    return 0;
}
```