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. I RIMANENTI TRE VALGONO 4 PUNTI.**
- **ALLA FINE DELLA PROVA, RICONSEGNARE TUTTI I FOGLI UTILIZZATI.**
- **SI RICORDA CHE LO SCRITTO VALE PER TRE APPELLI CONSECUTIVI (COMPRESO L'APPELLO IN CUI LO SCRITTO È SOSTENUTO)**

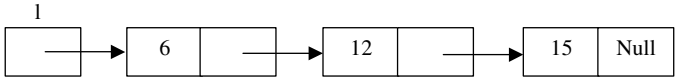
1) Scrivere una funzione ricorsiva che stampi su video una **v** composta da `N` asterischi, con `N` intero positivo dispari maggiore di 1. Per esempio, con `N`=5, la funzione stampa su video la **v** seguente

```
*       *
  *   *
    *
```

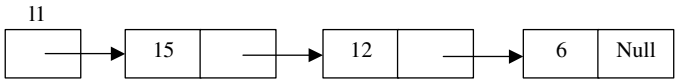
2) Sia data la struttura seguente

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

Scrivere una funzione che, data una lista `l` di strutture di tipo `elem` ordinata per valori crescenti del campo `val`, restituisca una nuova lista `l1` di strutture di tipo `elem` ordinata per valori decrescenti del campo `val`. Per esempio, data la lista



la funzione restituisce la lista



3) Scrivere una funzione che, data una matrice `m` di `n`×`k` interi ed un vettore `b` di `n` booleani, restituisce una matrice `m1` composta dalle righe della matrice `m` che hanno il corrispondente elemento in `b` a `false`. Per esempio, data la matrice `m` = $\begin{bmatrix} 1 & 2 & 3 \\ 0 & 3 & 3 \\ 3 & 4 & 4 \\ 1 & 5 & 3 \end{bmatrix}$ ed il vettore `b` = $\begin{bmatrix} false \\ true \\ false \\ true \end{bmatrix}$ la funzione restituisce la matrice `m1` = $\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 4 \end{bmatrix}$

4) Data la rappresentazione $(342A)_{16}$ in base 16, trasformarla in base 4.
Data la rappresentazione in complemento a due $(10001010)_{compl2}$, 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 a;
  public:
     A(int n=0) { a=n; cout << "A : a = " << a << endl;}
     virtual void f() { cout << "A::f() a = " << a << endl;}
     virtual ~A(){cout << "via A" << endl;}
};

class B: public A
{
   protected:
     int b;
  public:
     B(int n=4): A(n) { b=a; cout << "B: b = " << b << endl; }
     void f() { cout << "B::f() x = " << b << endl;}
     ~B(){cout << "via B" << endl;}
};

class C: public B
{
public:
      A aa;
      C(int n=0) : B(n) { a++; b++; cout << "C: a = " << a << endl; }
      void f() { cout << "C::f() b = " << b << endl;}
      ~C(){cout << "via C" << endl;}
   };
```

```
int main(){
   A* pa = new C(3);
   pa->f();
   B* pb = new C(2);
   pb->f();
   delete pa;
   delete pb;
   return 0;
}
```

6) Si mostri l'uscita a video del programma C++ seguente con ingresso -1, 0, +1.

```
#include<iostream>
using namespace std;
class Ecc {
  public:
       Ecc() {cout << "Ecc. ";}
       virtual void mess() = 0;
};

class Pos : public Ecc {
  public:
   Pos() {cout << "Pos." << endl;}
   void mess() {cout << "Esci Pos." << endl;}
};

class Zero : public Ecc {
  public:
   Zero() {cout << "Zero" << endl;}
   void mess() {cout << "Esci Zero" << endl;}
};
```

```
void check(int x) {
   if (x>0) throw Pos();
   if (x==0) throw Zero();
   throw x;
}

void f(int x) {
     try
     { check(x); }
     catch(int i){ cout << "Neg" << endl; }
     cout << x + 100 << endl;
}

int main () {
   int x;
   cin >> x;
   try{ f(x);}
   catch(Ecc& e){e.mess();}
   return 0;
}
```