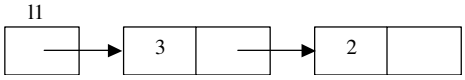
NOTE SULLO SVOLGIMENTO DELLA PROVA SCRITTA:

- **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) Scrivere una funzione ricorsiva che, dato un vettore *v* di interi, restituisca il numero di elementi dispari contenuti nel vettore.
   Per esempio, se *v* = [10, 21, 30, 43, 55], la funzione restituirà 3; se *v* = [10, 2, 30], la funzione restituirà 0.

2) Scrivere una funzione che, dato un intero positivo *val*, restituisca una lista che contenga, disposti in ordine decrescente, tutti i numeri interi *x*, con $1<x<val$, divisori di *val*. Nell'esempio seguente, sia 6 il numero intero positivo, la funzione restituisce la lista seguente:



3) Scrivere una funzione che, dato un vettore di interi, restituisca una matrice quadrata che ha tutti gli elementi uguali a 0 eccetto gli elementi sulla diagonale che sono uguali agli elementi del vettore.
   Per esempio, se *v* = [1, 2, 3], la matrice risultante sarà:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

   Se *v* = [70], la matrice risultante sarà:
   [70]

4) Data la rappresentazione $(207)_9$ in base 9, trasformarla in base 7.

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

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

class A{
public:
    int x;
    A() {  x=5;  cout << "nuovo A"  << endl; }
    void f() {
        cout << "A:f()" << endl << "x=" <<  x << endl;}
    ~A() {  cout << "via A"  << endl; }
};

class B: public A{
public:
     B() {  x=7;  cout << "nuovo B"  << endl; }
     virtual void f() { cout << "B:f()" << endl << "x=" <<  x << endl; }
     ~B() {  cout << "via B"  << endl; }
} ;

class C: public B{
    int x;
public:
     C() {  x=2;  cout << "nuovo C"  << endl; }
     void f() { cout << "C:f()" << endl << "x=" <<  x << endl; }
     ~C() {  cout << "via C"  << endl; }
};
```

```
class D: public A{
     int x;
public:
      D() {  x=20;  cout << "nuovo D"  << endl; }
      ~D() {  cout << "via D"  << endl; }
};


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

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

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

int fun (int a){
        cout << a << endl;
        try {       if (a==1) throw 1;
                    if (a==0) throw 'c';
                    cout << a+1 << endl;
        }
        catch(char) { cout << "char da fun" << endl;}
        cout << "fine fun" << endl;
        return a+1;
        };
```

```
int main(){
   int x;
   cin >> x;
   try {
      cout << x << endl;
      if (x==1) throw 'c';
      cout << fun(x) << endl;}
   catch(char) { cout << "char da main" << endl;}
   catch(int) { cout << "int da main" << endl;}
   catch (...) { cout << "default da main" << endl;}
   cout << "fine main" << endl;
   return 0;
}
```