NOTE SULLO SVOLGIMENTO DELLA PROVA SCRITTA:

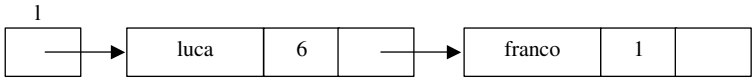
- **SPEGNERE I TELEFONINI;**
- **SCRIVERE IL PROPRIO NOME, COGNOME E NUMERO DI MATRICOLA SU OGNI FOGLIO UTILIZZATO;**
- **NON È POSSIBILE CONSULTARE NESSUN TIPO DI MATERIALE;**
- **NON È POSSIBILE UTILIZZARE CALCOLATRICI;**
- **PRIMA DI SCRIVERE LA SOLUZIONE DELL'ESERCIZIO, INSERIRE IL NUMERO DI ESERCIZIO 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, date due stringhe `s1` e `s2`, restituisce `true` se `s1` coincide con la parte iniziale di `s2`; altrimenti restituisce `false`. Nella scrittura della funzione non si possono utilizzare le funzioni di libreria che manipolano le stringhe. Per esempio, date le stringhe "croce" e "crocevia", la funzione restituisce `true`; date le stringhe "fine" e "finta", la funzione restituisce `false`.

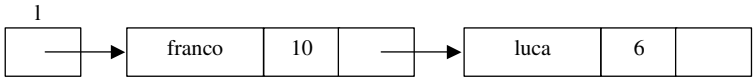
2) Sia data la struttura seguente

```
struct elem{char* nome; int cont; elem* pun;};
```

Scrivere una funzione che, data una lista `l` (non ordinata) di elementi di tipo `elem`, ed una stringa `s`, trova l'elemento con il campo `nome` uguale alla stringa `s` (si supponga che l'elemento sia unico), decrementa il corrispondente campo `cont` e, se il campo `cont` decrementato è uguale a zero, assegna 10 al campo `cont` e sposta l'elemento in testa alla lista. La funzione restituisce `true` se trova l'elemento, `false` altrimenti. Per esempio, sia `l` la lista seguente:



Se la funzione viene chiamata passando la stringa "franco" come parametro, la lista è modificata come:



3) Scrivere una funzione che, data una matrice `m` di $n \times k$ interi ed un vettore `v` di k interi, restituisce il prodotto della matrice `m` per il vettore `v` (si ricorda che il prodotto è un vettore di n interi). Per esempio, data la matrice `m` ed il vettore `v` seguenti:

$$\texttt{m} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & 6 \end{bmatrix} \text{ ed il vettore } \texttt{v} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

la funzione restituisce il vettore [14 20 26 32].

4) Data la rappresentazione $(45AF)_{16}$ in base 16, trasformarla in base 4.
Data la rappresentazione in complemento a due $(11101010)_{compl2}$, 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 x;
  public:
     A(int n) { x=n; cout << "A : x = " << x << endl;}
     void f() { cout << "A::f() x = " << x << endl;}
     ~A(){cout << "via A" << endl;}
};

class B: public A
{ protected:
     int x;
  public:
     B(int n=4): A(n-2) { x=n; cout << "B: x = " << x << endl; }
     virtual void f() { cout << "B::f() x = " << x << endl;}
     ~B(){cout << "via B" << endl;}
};

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

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

6) Si mostri l'uscita a video del programma C++ seguente con ingresso -5, 0, +5.

```
#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 << "Rec. Pos." << endl;}
};

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

void g(int x) {
   if (x<0) throw Neg();
   if (x>0) throw Pos();
   cout << x + 5 << endl;
}
```

```
void f(int x) {
    try
    { g(x); }
    catch(Ecc& ecc){ ecc.mess();}
    cout << x + 10 << endl;
}

int main () {
   int x;
   cin >> x;
   try{ f(x);}
   catch(...){cout <<  "catch generico " << endl;}
   cout << x + 100 << endl;
   return 0;
}
```