## 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 stampi su uscita standard un triangolo rettangolo composto di asterischi. I due cateti del triangolo contengono lo stesso numero `n` di asterischi. Nell'esempio seguente `n = 5`:

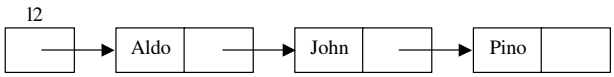
```
*
* *
* * *
* * * *
* * * * *
```

2) Sia data la struttura seguente

```
struct elem {char nome[20]; elem* pun;};
```

Scrivere una funzione che, data una lista `l1` non ordinata, restituisce una lista `l2` ordinata per valori alfabeticamente crescenti del campo `nome`. Eventuali elementi duplicati nella lista `l1` non vengono riportati nella lista `l2`. Per esempio, data la lista `l1` seguente


la lista `l2` restituita è la seguente




3) Scrivere una funzione che, dati due vettori `v1` e `v2` di, rispettivamente, `r` e `c` elementi, restituisce la matrice `m` di `r` righe e `c` colonne dove ogni elemento `m[i,j] = v1[i]*v2[j]`.
Per esempio, dati i due vettori `v1 = [1,2,3]` e `v2 = [4,5]`, la matrice `m` restituita dalla funzione è la seguente

```
 4  5
 8 10
12 15
```

4) Data la rappresentazione $(FA34)_{16}$ in base 16, trasformarla in base 2.
Data la rappresentazione in complemento a due $(11000111)_{compl2}$, 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;}
    virtual void f1() {cout << "A:f1() " << n << endl;}
    void f2() {cout << "A:f2() " << n << endl;}
} ;

class B: public A {
protected:   int n;
public:
   B() : A(3) { n = 10; cout << "B = " << n << endl;}
   void f1() {cout << "B:f1() " << n << endl;}
   virtual void f2() {cout << "B:f2() " << n << endl;}
};
```

```
class C: public B {
public:
   B b;
   C() {n++; cout << "C = " << n << endl;}
   C(int x) {n=x; cout << "C = " << n << endl;}
};


int main(){
    A* pa = new C;
    pa->f1();
    pa->f2();
    B* pb = new C(2);
    pb->f1();
    pb->f2();
    return 0;
}
```

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

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

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

class E0: public E {
 public:
    E0() { c = '0'; }
};

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

void f1(int i) {
   try {
   if( i==1 )
     throw E1();
   }
   catch(E1& e) {
      e.f();
    }
}
```

```
int f(int i) {
   try
   {
       f1(i);
       if (i == 0)
          throw E0();
       else
           if (i == 1)
              throw E1();
           else throw E();
   }
   catch(E0& e) {
     e.f();
    }
    return i;
}

int main() {
    try  { cout << f(0) << endl; }
    catch (E& e) {cout << "Ecc. Main" << endl;}
    try  { cout << f(1) << endl; }
    catch (E& e) {cout << "Ecc. Main" << endl;}
    try  { cout << f(2) << endl; }
    catch (E& e) {cout << "Ecc. Main" << endl;}
    return 0;
}
```