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

```
#include <iostream>
using namespace std;
template<class T1, class T2, int N>
class B
{    T1 a[N];
     T2 b[N];
     int size;
     int num;
public:
      B() { size = N; num = 0;}
      void p(T1 x, T2 y) { a[num] = x; b[num++] = y; }
      void g(T1* r1, T2* r2) { *r1 = a[--num]; *r2 = b[num]; }
      void print ()
      {   for (int i = 0 ; i < num ; i++ )
            cout << a[i] << " " << b[i] << endl;
      }
};

template<class T1, class T2>
void fun1 (T1 x, T2 y)
{   static B<T1, T2, 5> b;
    b.p (x,y);
    if ( x < y )
    {
        b.g(&x,&y);
        cout << x << ' ' << y << endl;
    }
    else
        b.print();
}
```

```
int main()
{   cout << "(1)" << endl;
    fun1 (3.2, 3);
    cout << endl << "(2)" << endl;
    fun1 (4.1, 4);
    cout << endl << "(3)" << endl;
    fun1 (3.8, 3);
    cout << endl << "(4)" << endl;
    fun1 (2.0, 14);
    cout << endl << "(5 )" << endl;
    fun1 (11.0, 7);
    return 0;
}
```

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

```
#include <iostream>
using namespace std;
class A
{
protected:
      int i;
public:
      A(){ i = 0; cout << "A" << endl;}
      void virtual f() = 0;
      void stampa(){ cout << "A " << i << endl;}
};

class B: public A
{     int j;
public:
      B(){ j = 1; cout << "B" << endl;}
      void f() { cout << j << ' ' << i << endl;}
      void stampa(){ cout << "B " << j << endl;}
};

class C: public A
{
protected:
      int z;
      B b;
public:
      C(){ z = 2; cout << "C"<< endl;}
      void f() { cout << z << ' ' << i << endl;}
};

class D: public C
{
protected:
      int z;
public:
      D(){ z = 3; cout << "D" << endl;}
      void stampa(){ cout << "D " << z << ' ' << C::z << endl;}
};
```

```
int main()
{
   cout << "(1)" << endl;
   D *d=new D;
   B *b=new B;
   A *a=d;
   cout << endl << "(2)" << endl;
   d->f();
   d->stampa();
   cout << endl << "(3)" << endl;
   a->f();
   a->stampa();
   cout << endl << "(4)" << endl;
   b->f();
   b->stampa();
   a = b;
   cout << endl << "(5)" << endl;
   a->f();
   a->stampa();
   return 0;
}
```

3) Scrivere la rappresentazione in base 2 dei numeri $(6532)_8$ e $(FA4A)_{16}$.
Scrivere la rappresentazione in complemento a due (utilizzando 8 cifre) del numero intero –106.

4) Scrivere una funzione ricorsiva che dato un vettore di interi, restituisce true se il vettore è ordinato per valori crescenti, false altrimenti. Il vettore è passato come argomento alla funzione. La funzione può essere chiamata con vettori di lunghezza diversa.

5) Scrivere una funzione che data una lista di elementi di tipo elem ed un intero x, elimina tutte le occorrenze di x dalla lista. La lista e l'intero x sono passati come argomento alla funzione. Si assuma che la struttura elem sia definita nel modo seguente:

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

6) Dato un vettore di interi ordinato per valori crescenti ed un intero x, scrivere una funzione che inserisce x nel vettore mantenendo il vettore ordinato (l'elemento di valore maggiore viene eliminato). Il vettore e l'intero x sono passati come argomento alla funzione. La funzione può essere chiamata con vettori di lunghezza diversa.