**Laurea in Informatica – Programmazione ad Oggetti – Appello d'Esame 24/6/2020**

**Esercizio 1**

```
class A {
public:
  virtual ~A() {}
};

class B: public A {
public:
  virtual bool m() const =0;
};

class C: public B {
public:
  bool m() const {return true;}
};

class D: virtual public A {
protected:
  D() {}
};

class E: public D {
public:
  E& operator=(const E& x) {
    D::operator=(x);
    return *this;
  }
};
```

Si consideri il codice sopra. Selezionare quali delle seguente affermazioni sono **VERE**:

1. `A` è una classe base polimorfa `A` **V**
2. Il costruttore di copia di `A` non è pubblicamente disponibile **F**
3. `A` è una classe base astratta; **F**
4. Il costruttore senza parametri di `A` è polimorfo **F**
5. `B` è una classe derivata astratta **V**
6. `B` è una classe polimorfa **V**
7. L'assegnazione di `B` è virtuale pura **F**
8. `B` è una classe virtuale pura **F**
9. `C` è una sottoclasse concreta di `B` **V**
10. `C` è una classe polimorfa **V**
11. `C` è una classe che non permetta la costruzione pubblica dei suoi oggetti **F**
12. Nella classe `C` il metodo `m()` è overloaded **F**
13. `D` è una classe che non permette di costruire pubblicamente i suoi oggetti **V**
14. `D` è una classe che permette la costruzione di oggetti di `D` che siano sottooggetti di altri oggetti **V**
15. Se `d` è un oggetto di tipo `D` allora in un main() la chiamata `d->m();` non compila **F**
16. `D` ha il costruttore di copia protetto **F**
17. `E` è una classe derivata indirettamente da `D` **F**
18. L'assegnazione ridefinita di `E` ha comportamento identico a quello dell'assegnazione standard di `E` **V**
19. Se `c` è un oggetto di tipo `C` allora in un main lo statement `E e(c);` compila correttamente **F**
20. `E` è una classe derivata indirettamente da `D` **F**
21. Se `d` è un oggetto di tipo `D` e `c` è un oggetto di tipo `C` allora in un main lo statement `c=e;` compila correttamente **V**

**Esercizio 2**

Siano `A`, `B`, `C` e `D` distinte classi polimorfe e si considerino le seguenti definizioni.

```
template<class X>
X* fun(X& ref) { return &ref; }

int main() {
  B b;
  fun<A>(b);
  B* p = new D();
  C c;
  if (dynamic_cast<B*>(fun<A>(c))) cout << "bianco ";
  else cout << "nero ";
  if( !(dynamic_cast<D*>(new B())) ) cout << "rosso ";
}
```

Si supponga che:

1. il `main()` **compili correttamente ed esegua senza provocare errori a run-time**;
2. l'esecuzione del `main()` provochi in output su `cout` la stampa `nero rosso`.

Sotto queste ipotesi, selezionare per ognuna delle seguente relazioni di sottotipo $X \leq Y$ quali sono **sicuramente vere**:

- $A \leq B$
- $A \leq C$
- $A \leq D$
- $B \leq A$ **V**
- $B \leq C$
- $B \leq D$
- $C \leq A$ **V**
- $C \leq B$
- $C \leq D$
- $D \leq A$ **V**
- $D \leq B$ **V**
- $D \leq C$

**Esercizio 3**

```
class A {
public:
  virtual void f() const {cout <<" A::f ";}
  virtual void g() {cout <<" A::g "; m(); }
  virtual void h() {cout <<" A::h "; f();}
  void m() {cout <<" A::m "; f();}
  virtual A* n() {cout <<" A::n "; return this;}
};

class C: public A {
public:
  virtual void f() {cout <<" C::f ";}
  void g() const {cout <<" C::g "; g();}
  void m() {cout <<" C::m "; g(); f();}
};
```

```
class B: public A {
public:
  virtual void f() const {cout <<" B::f ";}
  void g() {cout <<" B::g "; A::n();}
  virtual void m() {cout <<" B::m "; f();}
  A* n() {cout <<" B::n "; return this;}
};


class D: public B {
public:
  B* n() {cout <<" D::n "; return this;}
protected:
  void r() {cout <<" D::r ";}
public:
  void m() {cout <<" D::m "; f(); r();}
};
```

```
A* q1 = new D(); A* q2 = new B(); A* q3 = new C(); B* q4 = new D(); const A* q5 = new C();
```

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio **UNA RISPOSTA PER OGNI LINEA**:

- **NC** se l'istruzione **N**on **C**ompila correttamente;
- **ERT** se l'istruzione compila correttamente ma la sua esecuzione provoca un **E**rrore a **R**un **T**ime;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **precisamente** la stampa che l'esecuzione produce in output su `cout`; se provoca **N**essuna **S**tampa allora si scriva **NS**.
- **??** se non si è in grado di fornire una risposta.

```
q1->f();
q1->g();
q2->h();
q2->m();
q3->g();
q3->h();
q4->m();
q4->g();
(q3->n())->m();
(q3->n())->n()->f();
(q4->n())->m();
(q5->n())->f();
(dynamic_cast<B*>(q1))->m();
(static_cast<C*>(q2))->g();
(static_cast<B*>(q3->n()))->f();
```

```
B::f
B::g  A::n
A::h  B::f
A::m  B::f
A::g  A::m  A::f
A::h  A::f
D::m  B::f  D::r
B::g  A::n
A::n  A::m  A::f
A::n  A::n  A::f
D::n  A::m  B::f
NC
D::m  B::f  D::r
ERT
A::n  A::f
```