```
class B {
                                                            class C: public B {
protected:
                                                            public:
                                                              virtual void g() const {cout <<"C::g ";}</pre>
 virtual void h() {cout<<"B::h ";}</pre>
public:
                                                              void k() {cout << "C::k "; B::n();}</pre>
 virtual void f() {cout <<"B::f "; g(); h();}</pre>
                                                              virtual void m() {cout << "C::m "; g(); h();}</pre>
 virtual void g() const {cout <<"B::g";}</pre>
                                                              B* n() {cout <<"C::n "; return this;}</pre>
 virtual void k() {cout <<"B::k "; h(); m(); }</pre>
 void m() {cout <<"B::m "; g(); h();}</pre>
                                                                           P1->60
  virtual B* n() {cout <<"B::n "; return this;}</pre>
class D: public B {
                                                            class E: public C {
protected:
                                                            protected:
 void h() {cout <<"D::h ";}</pre>
                                                              void h() {cout <<"E::h ";}</pre>
public:
                                                            public:
 virtual void g() {cout <<"D::q ";}</pre>
                                                              void m() {cout <<"E::m "; q(); h();}</pre>
 void k() const {cout <<"D::k "; k();}</pre>
                                                              C* n() {cout <<"E::n "; return this;}</pre>
 void m() {cout <<"D::m "; g(); h();}</pre>
                                                            };
const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = new E(); B* p5 = new E();
```

- NON COMPILA se la compilazione dell'istruzione provoca un errore;
- UNDEFINED BEHAVIOUR se l'istruzione compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **chiaramente** la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
01: p1->g();
02: (p1->n())->g();
03: p2->f();
04: p2->m();
05: (static_cast<D*>(p2))->k();
06: p3->k();
07: p3->f();
08: (p3->n())->m();
09: (p3->n())->n()->g();
10: (static_cast<C*>(p3->n()))->g();
11: (p4->n())->m();
12: p5->g();
13: p5->k();
14: (dynamic_cast<C*>(p5))->m();
```

```
class B {
                                                          class C: public B {
protected:
                                                          public:
                                                            virtual void g() const {cout <<"C::g ";}</pre>
 virtual void h() {cout<<"B::h ";}</pre>
public:
                                                            void k() {cout << "C::k "; B::n();}</pre>
 virtual void f() {cout <<"B::f "; g(); h();}</pre>
                                                            virtual void m() {cout << "C::m "; g(); h();}</pre>
 virtual void g() const {cout <<"B::g ";}</pre>
                                                            B* n() {cout << "C::n "; return this;}</pre>
 virtual void k() {cout <<"B::k "; h(); m(); }</pre>
 void m() {cout <<"B::m "; g(); h();}</pre>
                                                                           (e1->NO) ->GC).
 virtual B* n() {cout <<"B::n "; return this;}</pre>
                                     THIS NON CONS
class D: public B {
                                                          class E: public C {
                                       P1 51
                                                         protected:
protected:
 void h() {cout <<"D::h ";}</pre>
                                                           void h() {cout <<"E::h ";}</pre>
public:
                                                          public:
 virtual void g() {cout <<"D::q ";}</pre>
                                                           void m() {cout <<"E::m "; q(); h();}</pre>
                                              S O
  void k() const {cout <<"D::k "; k();}</pre>
                                                            C* n() {cout <<"E::n "; return this;}</pre>
  void m() {cout <<"D::m "; g(); h();}</pre>
                                                      NON COMPILA
const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = new E(); B* p5 = new E();
```

- NON COMPILA se la compilazione dell'istruzione provoca un errore;
- UNDEFINED BEHAVIOUR se l'istruzione compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **chiaramente** la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
01: p1->g();
02: (p1->n())->g();
03: p2->f();
04: p2->m();
05: (static_cast<D*>(p2))->k();
06: p3->k();
07: p3->f();
08: (p3->n())->m();
09: (p3->n())->n()->g();
10: (static_cast<C*>(p3->n()))->g();
11: (p4->n())->m();
12: p5->g();
13: p5->k();
14: (dynamic_cast<C*>(p5))->m();
```

```
class B {
                                                             class C: public B {
protected:
                                                             public:
 virtual void h() {cout<<
                                                               virtual void g() const {cout <<::g
public:
                                                               void k() {cout << "C::k "; B::n();}</pre>
  virtual void f() {cout << "B::f" "; g(); h();}</pre>
                                                               virtual void m() {cout << "C::m "; g(); h();}</pre>
  virtual void g() const {cout <<"B::g ";}</pre>
                                                               B* n() {cout << "C::n "; return this;}</pre>
  virtual void k() {cout <<"B::k "; h(); m(); }</pre>
  void m() {cout <<"B::m "; g(); h();}</pre>
  virtual B* n() {cout <<"B::n "; return this;}</pre>
                                                                             P2->F();
class D: public B {
                                                             class E: public C {
                                                             protected:
protected:
  void h() {cout <<"D::h ";}</pre>
                                                               void h() {cout <<"E::h ";}</pre>
public:
                                                             public:
 virtual void g() {cout <<"D::q ";}</pre>
                                                              void m() {cout <<"E::m "; g(); h();}</pre>
  void k() const {cout <<"D::k "; k();}</pre>
                                                               C* n() {cout <<"E::n "; return this;}</pre>
 void m() {cout <<"D::m "; g(); h();}</pre>
                                                             };
const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = new E(); B* p5 = new E();
```

- NON COMPILA se la compilazione dell'istruzione provoca un errore;
- UNDEFINED BEHAVIOUR se l'istruzione compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **chiaramente** la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
01: p1->g();
02: (p1->n())->g();
03: p2->f();
04: p2->m();
05: (static_cast<D*>(p2))->k();
06: p3->k();
07: p3->f();
08: (p3->n())->m();
09: (p3->n())->n()->g();
10: (static_cast<C*>(p3->n()))->g();
11: (p4->n())->m();
12: p5->g();
13: p5->k();
14: (dynamic_cast<C*>(p5))->m();
```

```
// SOLUZIONE
01: B::g
02: NON COMPILA
03: B::f C::g B::h
04: B::m C::q B::h
05: UNDEFINED BEHAVIOUR
06: B::k D::h B::m B::g D::h
07: B::f
        B::g D::h
08: B::n B::m B::g
                    D::h
09: B::n
        B::n
              B::q
10: B::n
         B::q
11: E::n B::m C::g E::h
12: C::g
13: C::k B::n
14: E::m C::g E::h
```

```
class B {
                                                            class C: public B {
protected:
                                                            public:
                                                              virtual void g() const {cout << C::g
 virtual void h() {cout< B::h
public:
                                                              void k() {cout << "C::k "; B::n();}</pre>
  virtual void f() {cout <<"B::f "; g(); h();}</pre>
                                                              virtual void m() {cout << "C::m "; g(); h();}</pre>
  virtual void g() const {cout <<"B::g ";}</pre>
                                                              B* n() {cout << "C::n "; return this;}</pre>
  virtual void k() {cout <<"B::k "; h(); m(); }</pre>
  yoid m() {cout << "B::m "; g(); h();}</pre>
                                                                                  P2 -> MC).
  wirtual B* n() {cout << "B::n "; return this;}
        MO TROUBL
class D: public B {
                                                            class E: public C {
                                                            protected:
protected:
  void h() {cout <<"D::h ";}</pre>
                                                              void h() {cout <<"E::h ";}</pre>
public:
                                                            public:
  virtual void g() {cout <<"D::q ";}</pre>
                                                              void m() {cout <<"E::m "; g(); h();}</pre>
  void k() const {cout <<"D::k "; k();}</pre>
                                                              C* n() {cout <<"E::n "; return this;}</pre>
 void m() {cout <<"D::m "; g(); h();}</pre>
                                                            };
const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = new E(); B* p5 = new E();
```

- NON COMPILA se la compilazione dell'istruzione provoca un errore;
- UNDEFINED BEHAVIOUR se l'istruzione compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **chiaramente** la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
01: p1->g();
02: (p1->n())->g();
03: p2->f();
04: p2->m();
05: (static_cast<D*>(p2))->k();
06: p3->k();
07: p3->f();
08: (p3->n())->m();
09: (p3->n())->n()->g();
10: (static_cast<C*>(p3->n()))->g();
11: (p4->n())->m();
12: p5->g();
13: p5->k();
14: (dynamic_cast<C*>(p5))->m();
```

```
// SOLUZIONE
01: B::g
02: NON COMPILA
03: B::f C::g B::h
04: B::m C::q B::h
05: UNDEFINED BEHAVIOUR
06: B::k D::h B::m B::g D::h
07: B::f
        B::g D::h
08: B::n B::m B::g
                    D::h
09: B::n B::n
              B::q
10: B::n
         B::q
11: E::n B::m C::g E::h
12: C::g
13: C::k B::n
14: E::m C::g E::h
```

```
class B {
                                                             class C: public B {
protected:
                                                             public:
                                                               virtual void g() const {cout <<"C::g ";}</pre>
                                                                                                                  CON
 virtual void h() {cout<<"B::h ";}</pre>
                                                                                                                    STATIC
public:
                                                               void k() {cout <<"C::k "; B::n();}</pre>
 virtual void f() {cout <<"B::f "; g(); h();}</pre>
                                                               virtual void m() {cout << "C::m "; g(); h();}</pre>
                                                                                                                     CASO
 virtual void g() const {cout <<"B::g ";}</pre>
                                                               B* n() {cout << "C::n "; return this;}</pre>
 virtual void k() {cout <<"B::k "; h(); m(); }</pre>
 void m() {cout <<"B::m "; g(); h();}</pre>
                                                                     ( LSTATIC-CAST (D >) P2)) >16();
 virtual B* n() {cout <<"B::n "; return this;}</pre>
};
class D: public B {
                                                             class E: public C {
                                                             protected:
protected:
 void h() {cout <<"D::h ";}</pre>
                                                               void h() {cout <<"E::h ";}</pre>
public:
                                                             public:
 virtual void g() {cout <<"D::g ";}</pre>
                                                               void m() {cout <<"E::m "; q(); h();}</pre>
 void k() const {cout <<"D::k "; k();} void m() {cout <<"D::m "; g(); h();}
                                                               C* n() {cout <<"E::n "; return this;}</pre>
                                                             };
const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = new E(); B* p5 = new E();
```

- NON COMPILA se la compilazione dell'istruzione provoca un errore;
- UNDEFINED BEHAVIOUR se l'istruzione compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **chiaramente** la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
01: p1->g();
02: (p1->n())->g();
03: p2->f();
04: p2->m();
05: (static_cast<D*>(p2))->k();
06: p3->k();
07: p3->f();
08: (p3->n())->m();
09: (p3->n())->n()->g();
10: (static_cast<C*>(p3->n()))->g();
11: (p4->n())->m();
12: p5->g();
13: p5->k();
14: (dynamic_cast<C*>(p5))->m();
```

```
// SOLUZIONE
01: B::g
02: NON COMPILA
03: B::f C::g B::h
04: B::m C::q B::h
05: UNDEFINED BEHAVIOUR
06: B::k D::h B::m B::g D::h
07: B::f B::g D::h
08: B::n B::m B::g
                    D::h
09: B::n B::n
              B::q
10: B::n
        B::q
11: E::n B::m C::g E::h
12: C::g
13: C::k B::n
14: E::m C::g E::h
```

```
class B {
                                                              class C: public B {
protected:
                                                              public:
                                                                virtual void g() const {cout <<"C::g ";}</pre>
 virtual void h() {cout<<"B::h ";}</pre>
public:
                                                                void k() {cout << "C::k "; B::n();}</pre>
 virtual void f() {cout << "B::f "; g(); h();}</pre>
                                                                virtual void m() {cout << "C::m "; g(); h();}</pre>
 virtual void g() const {cout << B::g'";}</pre>
                                                                B* n() {cout << "C::n "; return this;}</pre>
 virtual void k() {cout << "B::k"; h(); m(); }
void m() {cout << "B::m"; g(); h();}</pre>
  virtual B* n() {coyt <<"B::n)"; return this;}</pre>
                                                                                        P3->K()
class D: public B
                                                              class E: public C {
 void h() {cout << (5); h); }
protected:
                                                              protected:
                                                                void h() {cout <<"E::h ";}</pre>
public:
                                                              public:
 virtual void q() {cout <<"D::q ";}</pre>
                                                                void m() {cout <<"E::m "; g(); h();}</pre>
 void k() const {cout <<"D::k "; k();}</pre>
                                                                C* n() {cout <<"E::n "; return this;}</pre>
  void m() {cout <<"D::m "; q(); h();}</pre>
                                                              };
const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = new E(); B* p5 = new E();
```

- NON COMPILA se la compilazione dell'istruzione provoca un errore;
- UNDEFINED BEHAVIOUR se l'istruzione compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **chiaramente** la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
01: p1->g();
02: (p1->n())->g();
03: p2->f();
04: p2->m();
05: (static_cast<D*>(p2))->k();
06: p3->k();
07: p3->f();
08: (p3->n())->m();
09: (p3->n())->n()->g();
10: (static_cast<C*>(p3->n()))->g();
11: (p4->n())->m();
12: p5->g();
13: p5->k();
14: (dynamic_cast<C*>(p5))->m();
```

```
class B {
                                                           class C: public B {
protected:
                                                           public:
 virtual void h() {cout<<"B::h ";}</pre>
                                                             virtual void g() const {cout <<"C::g ";}</pre>
                                                             void k() {cout << "C::k "; B::n();}</pre>
 virtual void f() {cout <<"B::f"; g(); h();}
                                                             virtual void m() {cout << "C::m "; g(); h();}</pre>
 virtual void g() const {cout << "B: g"}
                                                             B* n() {cout << "C::n "; return this;}</pre>
 virtual void k() {cout <<"B::k "; h(); m();</pre>
 void m() {cout <<"B::m "; g(); h();}</pre>
  virtual B* n() {cout <<"B::n "; return this;}</pre>
                                                                                    P3-2F()
class D: public B {
                                                           class E: public C {
                                                           protected:
protected:
 void h() {cout <</br>
                                                             void h() {cout <<"E::h ";}</pre>
public:
                                                           public:
 virtual void g() {cout <<"D::g ";}</pre>
                                                             void m() {cout <<"E::m "; g(); h();}</pre>
  void k() const {cout <<"D::k "; k();}</pre>
                                                             C* n() {cout <<"E::n "; return this;}</pre>
  void m() {cout <<"D::m "; g(); h();}</pre>
                                                           };
const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = new E(); B* p5 = new E();
```

- NON COMPILA se la compilazione dell'istruzione provoca un errore;
- UNDEFINED BEHAVIOUR se l'istruzione compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **chiaramente** la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
01: p1->g();
02: (p1->n())->g();
03: p2->f();
04: p2->m();
05: (static_cast<D*>(p2))->k();
06: p3->k();
07: p3->f();
08: (p3->n())->m();
09: (p3->n())->n()->g();
10: (static_cast<C*>(p3->n()))->g();
11: (p4->n())->m();
12: p5->g();
13: p5->k();
14: (dynamic_cast<C*>(p5))->m();
```

```
// SOLUZIONE
01: B::g
02: NON COMPILA
03: B::f C::g B::h
04: B::m C::q B::h
05: UNDEFINED BEHAVIOUR
06: B::k D::h B::m B::g D::h
07: B::f B::g D::h
08: B::n B::m B::g
                    D::h
09: B::n B::n
              B::q
10: B::n
        B::q
11: E::n B::m C::g E::h
12: C::g
13: C::k B::n
14: E::m C::g E::h
```

```
class B {
                                                               class C: public B {
protected:
                                                               public:
                                                                 virtual void g() const {cout <<"C::g ";}</pre>
 virtual void h() {cout<<"B::h ";}</pre>
public:
                                                                 void k() {cout << "C::k "; B::n();}</pre>
 virtual void f() {cout <<"B::f"; g(); h();}</pre>
                                                                 virtual void m() {cout << "C::m "; g(); h();}</pre>
 virtual void g() const {cout << "E::g" }
virtual void k() {cout << "B::k "; h(); m(); }</pre>
                                                                 B* n() {cout <<"C::n "; return this;}</pre>
                                                                                         (83->N)->M().
 void m() {cout <<"B::m-"; g(); h();}</pre>
 virtual B* n() {cout} << "B::n "; return this;}</pre>
class D: public B
                                                               class E: public C {
protected:
                                                               protected:
 void h() {cout <
                                                                 void h() {cout <<"E::h ";}</pre>
public:
                                                               public:
 virtual void q() {cout <<"D::q ";}</pre>
                                                                 void m() {cout <<"E::m "; g(); h();}</pre>
  void k() const {cout <<"D::k "; k();}</pre>
                                                                 C* n() {cout <<"E::n "; return this;}</pre>
 void m() {cout <<"D::m "; g(); h();}</pre>
                                                               };
const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = new E(); B* p5 = new E();
```

- NON COMPILA se la compilazione dell'istruzione provoca un errore;
- UNDEFINED BEHAVIOUR se l'istruzione compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **chiaramente** la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
01: p1->g();
02: (p1->n())->g();
03: p2->f();
04: p2->m();
05: (static_cast<D*>(p2))->k();
06: p3->k();
07: p3->f();
08: (p3->n())->m();
09: (p3->n())->n()->g();
10: (static_cast<C*>(p3->n()))->g();
11: (p4->n())->m();
12: p5->g();
13: p5->k();
14: (dynamic_cast<C*>(p5))->m();
```

```
// SOLUZIONE
01: B::g
02: NON COMPILA
03: B::f C::g B::h
04: B::m C::q B::h
05: UNDEFINED BEHAVIOUR
06: B::k D::h B::m B::g D::h
07: B::f B::g D::h
08: B::n B::m B::g
                    D::h
09: B::n B::n
              B::q
10: B::n
         B::q
11: E::n B::m C::g E::h
12: C::g
13: C::k B::n
14: E::m C::g E::h
```

```
class B {
                                                               class C: public B {
protected:
                                                               public:
                                                                 virtual void g() const {cout <<"C::g ";}</pre>
 virtual void h() {cout<<"B::h ";}</pre>
public:
                                                                 void k() {cout << "C::k "; B::n();}</pre>
 virtual void f() {cout << "B::f "; g(); h();</pre>
                                                                 virtual void m() {cout << "C::m "; g(); h();}</pre>
 virtual void g() const {cout < P::g
virtual void k() {cout << "B::k "; h();</pre>
                                                                 B* n() {cout <<"C::n "; return this;}</pre>
 void m() {cout <<"B::m "; g(); h();}</pre>
                                                                                 (P3 -> N) -> N () -> 6().
 virtual B* n() {cout <<"B::n "; return this;}</pre>
};
class D: public B {
                                                               class E: public C {
                                                              protected:
protected:
 void h() {cout <<"D::h ";}</pre>
                                                                 void h() {cout <<"E::h ";}</pre>
public:
                                                               public:
 virtual void g() {cout <<"D::g ";}</pre>
                                                                 void m() {cout <<"E::m "; g(); h();}</pre>
  void k() const {cout <<"D::k "; k();}</pre>
                                                                 C* n() {cout <<"E::n "; return this;}</pre>
 void m() {cout <<"D::m "; g(); h();}</pre>
                                                               };
const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = new E(); B* p5 = new E();
```

- NON COMPILA se la compilazione dell'istruzione provoca un errore;
- UNDEFINED BEHAVIOUR se l'istruzione compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **chiaramente** la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
01: p1->g();
02: (p1->n())->g();
03: p2->f();
04: p2->m();
05: (static_cast<D*>(p2))->k();
06: p3->k();
07: p3->f();
08: (p3->n())->m();
09: (p3->n())->n()->g();
10: (static_cast<C*>(p3->n()))->g();
11: (p4->n())->m();
12: p5->g();
13: p5->k();
14: (dynamic_cast<C*>(p5))->m();
```

```
// SOLUZIONE
01: B::g
02: NON COMPILA
03: B::f C::g B::h
04: B::m C::q B::h
05: UNDEFINED BEHAVIOUR
06: B::k D::h B::m B::g D::h
07: B::f B::g D::h
08: B::n B::m B::g
                    D::h
09: B::n B::n
              B::q
10: B::n
         B::q
11: E::n B::m C::g E::h
12: C::g
13: C::k B::n
14: E::m C::g E::h
```

```
class B {
                                                             class C: public B {
                                                             public:
protected:
                                                               virtual void g() const {cout << "C::g ";}</pre>
 virtual void h() {cout<<"B::h ";}</pre>
                                                               void k() {cout << "C::k "; B::n();}</pre>
public:
 virtual void f() {cout <<"B::f "; g() / h();}</pre>
                                                               virtual void m() {cout << "C::m "; g(); h();}</pre>
  virtual void g() const {cout << "E::g/";}</pre>
                                                               B* n() {cout <<"C::n "; return this;}</pre>
  virtual void k() {cout <<"B::k "; h(); m(); }</pre>
  void m() {cout <<"B::m "; g(); h();}</pre>
  virtual B* n() {cout <<"B::n "; return this;}</pre>
                                                             (static_cast<C*>(p3->n()))->g();
                               Ø
class D: public B {
                                                             class E: public C {
protected:
                                                             protected:
  void h() {cout <<"D::h ";}</pre>
                                                               void h() {cout <<"E::h ";}</pre>
public:
                                                             public:
  virtual void g() {cout <<"D::q ";}</pre>
                                                               void m() {cout <<"E::m "; g(); h();}</pre>
  void k() const {cout <<"D::k "; k();}</pre>
                                                               C* n() {cout <<"E::n "; return this;}</pre>
  void m() {cout <<"D::m "; q(); h();}</pre>
                                                             };
const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = new E(); B* p5 = new E();
```

- NON COMPILA se la compilazione dell'istruzione provoca un errore;
- UNDEFINED BEHAVIOUR se l'istruzione compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **chiaramente** la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
Il comportamento osservato è legato alla differenza fondamentale tra static_cast e dynamic_cast nei cast cross-hierarchy.

Analizziamo cosa accade:

1. p3 è un puntatore B* che punta a un oggetto D.

2. p3->n() chiama B::n() poiché D non lo sovrascrive, e restituisce this, un puntatore all'oggetto D con tipo statico B*.

3. static_cast<(C*>(p3->n()) tenta di convertire questo puntatore in un puntatore C*.

Questo è un cast cross-hierarchy (tra rami diversi della gerarchia) e produce un comportamento definito dal compilatore ma non dal linguaggio. Il static_cast non verifica la validità dell'operazione a runtime, consentendo la conversione pericolosa.
```

```
Quando chiamiamo g() attraverso questo puntatore, l'oggetto sottostante è ancora di tipo D, che eredita il metodo g() da B. La vtable utilizzata per la risoluzione del metodo virtuale è quella dell'oggetto effettivo (D), non quella del tipo del puntatore (C). Quindi viene chiamato B::g() anche se il puntatore è di tipo C*.

dynamic_cast<C*>(p3->n())->g() non compila perché dynamic_cast, a differenza di static_cast, esegue un controllo di tipo a runtime. Poiché D non è sottotipo di C (sono rami diversi dell'albero di ereditarietà), il cast restituirebbe nullptr. Il compilatore rileva che si sta tentando di chiamare un metodo su un puntatore potenzialmente nullo e genera un errore.

In sostanza, è una questione di sicurezza dei tipi: static_cast permette operazioni pericolose lasciando la responsabilità al programmatore, mentre dynamic_cast offre maggiore sicurezza rifiutando conversioni invalide.
```

```
class B {
                                                             class C: public B {
protected:
                                                             public:
                                                               virtual void g() const {cout < ("C::g":
 virtual void h() {cout<<"B::h ";}</pre>
public:
                                                               void k() {cout << "C::k "; B::n();}</pre>
  virtual void f() {cout <<"B::f "; g(); h();}</pre>
                                                               virtual void m() {cout << "C::m "; g(); h();}</pre>
  virtual void g() const {cout <<"B::g ";}</pre>
                                                               B* n() {cout <<"C::n "; return this;}</pre>
  virtual void k() {cout <<"B::k "; h(); m(); }</pre>
 void m() {cout << "B::m"; g(); h();}</pre>
                                                                        ( (4 → N &)) → M ().
  virtual B* n() {cout << "B::n "; return this;}
} ;
class D: public B {
                                                             class E: public C {
protected:
                                                             protected:
  void h() {cout <<"D::h ";}</pre>
                                                               void h() {cout <</pre>
public:
                                                             public:
  virtual void g() {cout <<"D::g ";}</pre>
                                                               void m() {cout <<"E::m "; g(); h();}</pre>
  void k() const {cout <<"D::k "; k();}</pre>
                                                               C* n() {cout <<"E::n "; return this;}</pre>
 void m() {cout <<"D::m "; g(); h();}</pre>
                                                                                  (t)
const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = new E(); B* p5 = new E();
```

- NON COMPILA se la compilazione dell'istruzione provoca un errore;
- UNDEFINED BEHAVIOUR se l'istruzione compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **chiaramente** la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
01: p1->g();
02: (p1->n())->g();
03: p2->f();
04: p2->m();
05: (static_cast<D*>(p2))->k();
06: p3->k();
07: p3->f();
08: (p3->n())->m();
09: (p3->n())->n()->g();
10: (static_cast<C*>(p3->n()))->g();
11: (p4->n())->m();
12: p5->g();
13: p5->k();
14: (dynamic_cast<C*>(p5))->m();
```

```
// SOLUZIONE
01: B::g
02: NON COMPILA
03: B::f C::g B::h
04: B::m C::q B::h
05: UNDEFINED BEHAVIOUR
06: B::k D::h B::m B::g D::h
07: B::f
        B::g D::h
08: B::n B::m B::g
                    D::h
09: B::n
         B::n
              B::q
10: B::n
         B::q
11: E::n B::m
              C::q E::h
12: C::g
13: C::k B::n
14: E::m C::g E::h
```

```
class B {
                                                              class C: public B {
protected:
                                                              public:
                                                                virtual void g() const {cout </ color="c".g"
  virtual void h() {cout<<"B::h ";}</pre>
public:
                                                                void k() {cout << "C::k "; B::n();
  virtual void f() {cout <<"B::f "; g(); h();}</pre>
                                                                virtual void m() {cout << "C::m "; g(); h();}</pre>
  virtual void g() const {cout <<"B::g ";}</pre>
                                                                B* n() {cout <<"C::n "; return this;}</pre>
  virtual void k() {cout <<"B::k "; h(); m(); }</pre>
  void m() {cout <<"B::m "; g(); h();}</pre>
  virtual B* n() {cout <<"B::n "; return this;}</pre>
class D: public B {
                                                              class E: public C {
                                                              protected:
protected:
  void h() {cout <<"D::h ";}</pre>
                                                                void h() {cout <<"E::h ";}</pre>
                                                              public:
public:
  virtual void g() {cout <<"D::g ";}</pre>
                                                                void m() {cout <<"E::m "; q(); h();}</pre>
  void k() const {cout <<"D::k "; k();}</pre>
                                                                C* n() {cout <<"E::n "; return this;}</pre>
 void m() {cout <<"D::m "; g(); h();}</pre>
const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = new E(); B* p5 = new E();
```

- NON COMPILA se la compilazione dell'istruzione provoca un errore;
- UNDEFINED BEHAVIOUR se l'istruzione compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **chiaramente** la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
01: p1->g();
02: (p1->n())->g();
03: p2->f();
04: p2->m();
05: (static_cast<D*>(p2))->k();
06: p3->k();
07: p3->f();
08: (p3->n())->m();
09: (p3->n())->n()->g();
10: (static_cast<C*>(p3->n()))->g();
11: (p4->n())->m();
12: p5->g();
13: p5->k();
14: (dynamic_cast<C*>(p5))->m();
```

```
// SOLUZIONE
01: B::g
02: NON COMPILA
03: B::f C::g B::h
04: B::m C::q B::h
05: UNDEFINED BEHAVIOUR
06: B::k D::h B::m B::g D::h
07: B::f B::g D::h
08: B::n B::m B::g
                    D::h
09: B::n B::n
              B::q
10: B::n
         B::q
11: E::n B::m C::g E::h
12: C::g
13: C::k B::n
14: E::m C::g E::h
```

```
class B {
                                                             class C: public B {
protected:
                                                            public:
                                                              virtual void g() const {cout <<"C::g ";}</pre>
 virtual void h() {cout<<"B::h ";}</pre>
public:
                                                              void k() {cout << "C::k "; B::n();}</pre>
                                                              virtual void m() {cout << "C::m, "; g(); h();}
  virtual void f() {cout <<"B::f "; g(); h();}</pre>
                                                              B* n() {cout << "C::n "; return his;}
  virtual void g() const {cout <<"B::g ";}</pre>
  virtual void k() {cout << "B::k "; h(); m(); }</pre>
  void m() {cout <<"B::m "; g(); h();}</pre>
                                                                                               PS -> WU.
  virtual B* n() {cout <<"B::n "; return this;}</pre>
class D: public B {
                                                             class E: public C {
                                                             protected:
protected:
 void h() {cout <<"D::h ";}</pre>
                                                               void h() {cout <<"E::h ";}</pre>
public:
                                                             public:
 virtual void g() {cout <<"D::q ";}</pre>
                                                               void m() {cout <<"E::m "; q(); h();}</pre>
  void k() const {cout <<"D::k "; k();}</pre>
                                                               C* n() {cout <<"E::n "; return this;}</pre>
 void m() {cout <<"D::m "; g(); h();}</pre>
const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = new E(); B* p5 = new E();
```

- NON COMPILA se la compilazione dell'istruzione provoca un errore;
- UNDEFINED BEHAVIOUR se l'istruzione compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **chiaramente** la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
01: p1->g();
02: (p1->n())->g();
03: p2->f();
04: p2->m();
05: (static_cast<D*>(p2))->k();
06: p3->k();
07: p3->f();
08: (p3->n())->m();
09: (p3->n())->n()->g();
10: (static_cast<C*>(p3->n()))->g();
11: (p4->n())->m();
12: p5->g();
13: p5->k();
14: (dynamic_cast<C*>(p5))->m();
```

```
class B {
                                                            class C: public B
protected:
                                                            public:
                                                              virtual void g() const {cout < ("C::g
  virtual void h() {cout<<"B::h ";}</pre>
                                                              void k() {cout << "C::k "; B::n();}</pre>
public:
  virtual void f() {cout <<"B::f "; g(); h();}</pre>
                                                              virtual void m() {cout << "C::m "; g(); h();}</pre>
  virtual void g() const {cout <<"B::g ";}</pre>
                                                              B* n() {cout << "C::n "; return this;}</pre>
  virtual void k() {cout <<"B::k "; h(); m(); }</pre>
  void m() {cout <<"B::m "; g(); h();}</pre>
                                                                        (DYNAMIC_CAST (CAST (P5) >170
  virtual B* n() {cout <<"B::n "; return this;}</pre>
class D: public B {
                                                            class E: public C {
                                                            protected:
protected:
                                                              void h() {cout ⟨<"E::b)';
  void h() {cout <<"D::h ";}</pre>
public:
                                                            public:
  virtual void g() {cout <<"D::g ";}</pre>
                                                              void m() {cout <<"<u>E::m.</u>"; g(); h();}
  void k() const {cout <<"D::k "; k();}</pre>
                                                              C* n() {cout <<"E::n "; return this;}</pre>
 void m() {cout <<"D::m "; g(); h();}</pre>
                                                            };
```

Le precedenti definizioni compilano correttamente. Per ognuno dei seguenti 14 statement in tabella con **numerazione da 01 a 14**, scrivere **chiaramente nel foglio 14 risposte con numerazione da 01 a 14** e per ciascuna risposta:

• NON COMPILA se la compilazione dell'istruzione provoca un errore;

const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = new E(); B* p5 = new E();

- UNDEFINED BEHAVIOUR se l'istruzione compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **chiaramente** la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
01: p1->g();
02: (p1->n())->g();
03: p2->f();
04: p2->m();
05: (static_cast<D*>(p2))->k();
06: p3->k();
07: p3->f();
08: (p3->n())->m();
09: (p3->n())->n()->g();
10: (static_cast<C*>(p3->n()))->g();
11: (p4->n())->m();
12: p5->g();
13: p5->k();
14: (dynamic_cast<C*>(p5))->m();
```

```
// SOLUZIONE
01: B::g
02: NON COMPILA
03: B::f C::g B::h
04: B::m C::q B::h
05: UNDEFINED BEHAVIOUR
06: B::k D::h B::m B::g D::h
07: B::f
         B::g D::h
08: B::n
         B::m
               B::g
                    D::h
09: B::n
         B::n
               B::q
10: B::n
         B::q
11: E::n B::m
              C::q E::h
12: C::g
13: C::k B::n
14: E::m C::g E::h
```

```
class B {
                                                            class C: public B {
protected:
                                                            public:
                                                              virtual void g() const {cout <<"C::g ";}</pre>
 virtual void h() {cout<<"B::h ";}</pre>
public:
                                                              void k() {cout << "C::k "; B::n();}</pre>
  virtual void f() {cout <<"B::f "; g(); h();}</pre>
                                                              virtual void m() {cout << "C::m "; g(); h();}</pre>
  virtual void g() const {cout <<"B::g ";}</pre>
                                                              B* n() {cout <<"C::n "; return this;}</pre>
  virtual void k() {cout <<"B::k "; h(); m(); }</pre>
 void m() {cout <<"B::m "; g(); h();}</pre>
  virtual B* n() {cout <<"B::n "; return this;}</pre>
                                                             COMPLIA SE CONST_CAST
} ;
                                               20
class D: public B {
                                                            class E: public C {
                                                            protected:
protected:
  void h() {cout <<"D::h ";}</pre>
                                                              void h() {cout <<"E::h ";}</pre>
public:
                                                            public:
  virtual void g() {cout <<"D::q ";}</pre>
                                                              void m() {cout <<"E::m "; g(); h();}</pre>
  void k() const {cout <<"D::k "; k();</pre>
                                                              C* n() {cout <<"E::n "; return this;}</pre>
 void m() {cout << "D::m "; g(); h____
                                                            };
const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = new E(); B* p5 = new E();
```

- NON COMPILA se la compilazione dell'istruzione provoca un errore;
- UNDEFINED BEHAVIOUR se l'istruzione compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **chiaramente** la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
01: p1->g();
02: (p1->n())->g();
03: p2->f();
04: p2->m();
05: (static_cast<D*>(p2))->k();
06: p3->k();
07: p3->f();
08: (p3->n())->m();
09: (p3->n())->n()->g();
10: (static_cast<C*>(p3->n()))->g();
11: (p4->n())->m();
12: p5->g();
13: p5->k();
14: (dynamic_cast<C*>(p5))->m();
```

```
// SOLUZIONE
01: B::g
02: NON COMPILA
03: B::f C::g B::h
04: B::m C::q B::h
05: UNDEFINED BEHAVIOUR
06: B::k D::h B::m B::g D::h
07: B::f B::g D::h
08: B::n B::m B::g
                    D::h
09: B::n B::n
              B::q
10: B::n
        B::q
11: E::n B::m C::g E::h
12: C::g
13: C::k B::n
14: E::m C::g E::h
```

```
class B {
                                                              class C: public B {
protected:
                                                              public:
                                                                virtual void g() const {cout <<"C::g ";}</pre>
 virtual void h() {cout<<"B::h ";}</pre>
public:
                                                                void k() {cout << "C::k "; B::n();}</pre>
 virtual void f() {cout <<"B::f "; g(); h();}</pre>
                                                                virtual void m() {cout << "C::m "; g(); h();}</pre>
 virtual void g() const {cout <<"B::g ";}</pre>
                                                                B* n() {cout << "C::n "; return this;}</pre>
  virtual void k() {cout <<"B::k "; h(); m(); }</pre>
 void m() {cout <<"B::m "; g(); h();}</pre>
                                                                      (static_cast<D*>(p2))\rightarrow k();
  virtual B* n() {cout <<"B::n "; return this;}</pre>
} ;
class D: public B {
                                                              class E: public C {
                                                              protected:
protected:
  void h() {cout <<"D::h ";}</pre>
                                                                void h() {cout <<"E::h ";}</pre>
public:
                                                              public:
 virtual void g() {cout <<"D::q ";}</pre>
                                                                void m() {cout <<"E::m "; g(); h();}</pre>
                                                                C* n() {cout <<"E::n "; return this;}</pre>
  void k() const {cout <<"D::k "; k();}</pre>
 void m() {cout <<"D::m "; g(); h();}</pre>
                                                              };
const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = new E(); B* p5 = new E();
```

- NON COMPILA se la compilazione dell'istruzione provoca un errore;
- UNDEFINED BEHAVIOUR se l'istruzione compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **chiaramente** la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

```
01: p1->g();
02: (p1->n())->g();
03: p2->f();
04: p2->m();
05: (static_cast<D*>(p2))->k();
06: p3->k();
07: p3->f();
08: (p3->n())->m();
09: (p3->n())->n()->g();
10: (static_cast<C*>(p3->n()))->g();
11: (p4->n())->m();
12: p5->g();
13: p5->k();
14: (dynamic_cast<C*>(p5))->m();
```