# [] (P1 -> N()) -> 6(). -> NC

## Esercizio ( Esercizio Cosa Stampa

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

```
P2>P1
Esercizio Cosa Stampa
 class B {
                                                             class C: public B {
 protected:
                                                             public:
                                                               virtual void g() const {cout < f"C::q
  virtual void h() {cout<√"B::h
 public:
                                                               void k() {cout << "C::k "; B::n();}</pre>
                             k"B::f " g(); h();}
  virtual void f() {cout <</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::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
```

[4] P2 >17();

```
Esercizio Cosa Stampa
 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()
void m() {cout <<"B::k "; h(); m(); }
void m() {cout <<"B::m ); g(); h();}
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>
   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::g E::h
12: C::g
13: C::k B::n
14: E::m C::g E::h
```

P3->K()

```
class B {
                                                                        class C: public B {
protected:
                                                                        public:
                                                                          virtual void g() const {cout <<"C::g ";}</pre>
  virtual void h() {cfut<<"B::h ";}</pre>
public:
                                                                          void k() {cout << "C::k "; B::n();}</pre>
 virtual void f() { dout << "B::f "; g(); h();}
virtual void g() const {cout << "B::g ");}
virtual void k() { const << "B::k "; h(); m(); }</pre>
                                                                          virtual void m() {cout << "C::m "; g(); h();}</pre>
                                                                          B* n() {cout <<"C::n "; return this;}</pre>
                           coat << "B::k "; h()
s::m "; g(); h();}</pre>
  void m() {cout < "B::m
  virtual B* n() {cout
                              <<"B::n "; return this; }
class D: public B
                                                                        class E: public C {
protected:
                                                                        protected:
                                                                          void h() {cout <<"E::h ";}</pre>
  void h() {cout <<"D::h</pre>
public:
                                                                        public:
  virtual void q() {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>
```

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:

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;

Esercizio Cosa Stampa

- 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
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
```

```
43 -> F()
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 g() const {cout < "B::g"; )
virtual void k() {cout << "B::k"; h(); m(); }</pre>
                                                                     virtual void m() {cout << "C::m "; g(); h();}</pre>
                                                                     B* n() {cout << "C::n "; return this;}</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() { fout << "D::h</pre>
                                                                     void h() {cout <<"E::h ";}</pre>
public:
                                                                   public:
  virtual void q() {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->q();
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
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
```

(P3->N())-> T()

```
Esercizio Cosa Stampa
class B {
                                                            class C: public B {
protected:
                                                            public:
  virtual void h() {cout<<"\mathbb{E}::h ";}</pre>
                                                             virtual void g() const {cout <<"C::g ";}</pre>
public:
                                                              void k() {cout << "C::k "; B::n();}</pre>
  virtual void m() {cout << "C::m "; g(); h();}</pre>
                                                             B* n() {cout <<"C::n "; return this;}</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
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
```

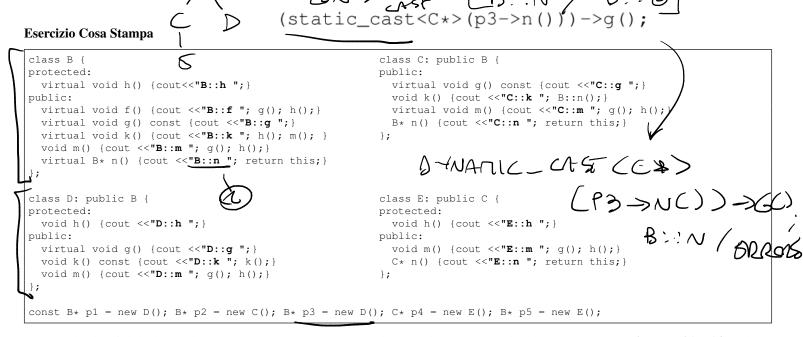
## (P3 ->NC)) -> N()->6()

```
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 "; })
virtual void k() {cout << "B::k "; h(); m(); }</pre>
                                                                   B* n() {cout << "C::n "; return this;}</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 "; 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
```



- 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
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
```

```
(P4 -> N()) -> T().
                                                                                                               (3
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();}</pre>
 virtual void f() {cout << "B::f "; 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;}
} ;
class D: public B {
                                                            class E: public C {
                                                            protected:
protected:
 void h() {cout <<"D::h</pre>
                                                              void h() {cout <
public:
                                                            public:
 virtual void q() {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>
```

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->q();
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
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 h() {cout<<"B::h ";}</pre>
                                                                      virtual void g() const {cout <<"C::g ";}</pre>
public:
                                                                      void k() {cout << "C::k "; B::n();}</pre>
 virtual void f() {cout <<"B::f"; g(); h();}
virtual void g() const {cout << B::g";}
virtual void k() {cout << "B::k"; h(); m(); }</pre>
                                                                      virtual void m() {cout << "C::m "; g(); h();}</pre>
                                                                      B* n() {cout << "C::n "; return this;}</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::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 "; q(); h();}</pre>
const B* p1 = new D(); B* p2 = new C(); B* p3 = new D(); C* p4 = 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::q
10: B::n B::g
11: E::n B::m C::g E::h
12: C::g
13: C::k B::n
14: E::m C::g E::h
```

### PS > K() Esercizio Cosa Stampa class B { class C: public B { protected: public: virtual void h() {cout<<"B::h ";}</pre> virtual void g() const {cout << "C::g ";</pre> public: void k() {cout <<"C::k "; B::n();}</pre> virtual void m() {cout <<"C::m "; g(); h();} virtual void f() {cout <<"B::f "; f(); h();} 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" " (); h();}</pre> virtual B\* n() {cout, << "B::n "; return this;}</pre> class D: public B { class E: public C { void h() {cout <<("E::h ";) protected: protected: void h() {cout <<"D::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();
```

```
// 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 ")
 virtual void h() {cout<<"B::h ";}</pre>
public:
                                                              void k() {cout << "C::k "; B::n();
                                                              virtual void m() {cout << "C::m "; g(); h();}</pre>
  virtual void f() {cout <<"B::f "; 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 </br>
  void h() {cout <<"D::h ";}</pre>
public:
                                                            public:
  virtual void g() {cout <<"D::g ";}</pre>
                                                              void m() {cout <
                                                                                 k"E::m ";)g(); h();}
                                                              C* n() {cout <<""; 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();
```

```
// 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
```