## Esercizio 3

```
class A {
                                                               class B: public A {
protected:
                                                               public:
  virtual void r() {cout<<" A::r ";}</pre>
                                                                 virtual void g() const {cout <<" B::g ";}</pre>
                                                                 virtual void m() {cout <<" B::m "; g(); r();}</pre>
public:
  virtual void g() const {cout <<" A::g ";}</pre>
                                                                 void k() {cout <<" B::k "; A::n();}</pre>
                                                                 A* n() {cout <<" B::n "; return this;}
  virtual void f() {cout << " A::f "; g(); r();}</pre>
  void m() {cout <<" A::m "; g(); r();}</pre>
 virtual void k() {cout <<" A::k "; r(); m(); }</pre>
  virtual A* n() {cout << " A::n "; return this;}</pre>
                                                                   \{static\_cast<C*>(p2)\}\rightarrow k();
};
                                                               class D: public B {
class C: public A {
protected:
                                                               protected:
  void r() {cout <<" C::r ";}</pre>
                                                                 void r() {cout <<" D::r ";}</pre>
public:
                                                               public:
                                                 SPIT
  virtual void g() {cout << " C::g ";}</pre>
                                                                 B* n() {cout <<" D::n "; return this;}</pre>
  void m() {cout <<" C::m "; g(); r();</pre>
                                                                 void m() {cout <<" D::m "; g(); r();}</pre>
            const {cout <<" C::k "(k();}
  void k()
                                                               };
                      CX &Z = NOW C ()-
A* p1 = new D(); A* p2 = new B(); A* p3 = new C(); B* p4 = new D(); const A* p5 = new C();
```

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

```
p1->k();

p2->f();

p2->m();

p3->k();

p3->f();

p5->g();

(p3->n())->m();

(p3->n())->m()->g();

(p4->n())->m();

(p5->n())->g();

(dynamic_cast<B*>(p1))->m();

(static_cast<C*>(p2))->k();
```

DIN-CAST (CONST\_CAST (PS->NC))

```
Esercizio 3
class A {
                                                             class B: public A {
protected:
                                                             public:
                                                               virtual void g() const {cout <<" B::g ";}</pre>
                                                                                                                   X 8HL
  virtual void r() {cout<<" A::r ";}</pre>
                                                               virtual void m() {cout <<" B::m "; g(); r();}</pre>
public:
  virtual void g() const {cout <<" A::g ";}</pre>
                                                              void k() {cout <<" B::k "; A::n();}</pre>
                                                              A* n() {cout <<" B::n "; return this;}
  virtual void f() {cout << " A::f "; g(); r();}</pre>
  void m() {cout <<" A::m "; g(); r();}</pre>
  virtual void k() {cout <<" A::k "; r(); m(); }</pre>
  virtual A* n() {cout << " A::n "; return this;}</pre>
                                                                                         (p5\rightarrow n())\rightarrow g(); \longrightarrow \bigwedge \subset
                                                             class D: public B {
class C: public A {
protected:
                                                             protected:
  void r() {cout <<" C::r ";}</pre>
                                                               void r() {cout <<" D::r ";}</pre>
public:
                                                             public:
  virtual void g() {cout <<" C::g ";}</pre>
                                                               B* n() {cout <<" D::n "; return this;}</pre>
  void m() {cout <<" C::m "; g(); r();}</pre>
                                                               void m() {cout <<" D::m "; g(); r();}</pre>
  void k() const {cout <<" C::k "; k();}</pre>
```

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

```
p1->k();

p2->f();

p3->k();

p3->f();

p5->g();

(p3->n())->n()->g();

(p4->n())->m();

(p5->n())->g();

(dynamic_cast<B+>(p1))->m();

(static_cast<C+>(p2))->k();
```

## Esercizio 3

```
class A {
                                                              class B: public A {
protected:
                                                              public:
  virtual void r() {cout<<" A::r ";}</pre>
                                                                 virtual void g() const {cout << " B::g ";}</pre>
                                                                 virtual void m() {cout <<" B::m "; g(); r();}</pre>
public:
                                                                 void k() {cout <<" B::k "; A::n();}</pre>
  virtual void g() const {cout <<" A::g ";}</pre>
                                                                 A* n() {cout <<" B::n "; return this;}
  virtual void f() {cout << " A::f "; g(); r();}</pre>
  void m() {cout <<" A::m "; g(); r();}</pre>
 virtual void k() {cout <<" A::k "; r(); m(); }</pre>
  virtual A* n() {cout << " A::n "; return this;}</pre>
                                                                          (dynamic_cast < B*>(p1)) \rightarrow m();
};
class C: public A {
                                                              class D: public B {
                                                              protected:
protected:
                                                                          {cout <<" D::
  void r() {cout <<" C::r ";}</pre>
                                                                 void r()
                                                              public:
public:
                                                                 B* n() {cout <<" D::n "; return this;}</pre>
  virtual void g() {cout << " C::g ";}</pre>
                                                                 void m() {cout <<" D::m "; g(); r();}</pre>
  void m() {cout << " C::m "; g(); r();}</pre>
  void k() const {cout <<" C::k "; k();}</pre>
};
               BXP1=NOW D();
A* p1 = new D(); A* p2 = new B(); A* p3 = new C(); B* p4 = new D(); const A* p5 = new C();
```

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

```
p1->k();
p2->f();
p2->m();
p3->k();
p3->f();
p3->f();
p5->g();
(p3->n())->m();
(p3->n())->m();
(p4->n())->m();
(p5->n())->g();
(dynamic_cast<B*>(p1))->m();
(static_cast<C*>(p2))->k();
```

## Esercizio 3

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

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

```
p1->k();
 ......
p2->f();
p2->m();
 ......
p3->k();
 p3->f(); .....
p5->q();
 .....
(p3-n())-m();
(p3-n())-n()-n();
(p4-n())-m();
(p5->n())->q();
(dynamic_cast<B*>(p1))->m(); .....
(static_cast<C*>(p2))->k();
```

```
Esercizio 3
class A {
                                                                       class B: public A {
protected:
                                                                      public:
                                                                         virtual void g() const {cout <<" B::q ";}</pre>
                                  A::r
   virtual void r() {cout<<
                                                                         virtual void m() {cout <<" B::m "; g(); r();}</pre>
public:
   virtual void g() const {cout <<" A::g ";}</pre>
                                                                         void k() {cout <<" B::k "; A::n();}</pre>
   virtual void f() {cout <<"_A::f "; g(); r();}</pre>
                                                                         A* n() {cout <<" B::n "; return this;}
  void m() {cout << " A::m "; g(); r();}
virtual void k() {cout << " A::k "; r(); m(); }</pre>
   virtual A* n() {cout <<" A::n "; return this;}</pre>
                                                                                                      p2\rightarrow m();
class C: public A {
                                                                      class D: public B {
protected:
                                                                      protected:
   void r() {cout <<" C::r ";}</pre>
                                                                         void r() {cout <<" D::r ";}</pre>
public:
                                                                      public:
   virtual void g() {cout <<" C::g ";}</pre>
                                                                         B* n() {cout <<" D::n "; return this;}</pre>
   void m() {cout << " C::m "; g(); r();}</pre>
                                                                         void m() {cout <<" D::m "; g(); r();}</pre>
   void k() const {cout <<" C::k "; k();}</pre>
                                                                      };
};
A* p1 = \text{new D()}; A* p2 = \text{new B()}; A* p3 = \text{new C()}; B* p4 = \text{new D()}; \text{const } A* p5 = \text{new C()};
```

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

```
p1->k();

p2->f();

p3->k();

p3->f();

p5->g();

(p3->n())->m();

(p3->n())->m();

(p4->n())->m();

(p5->n())->g();

(dynamic_cast<B+>(p1))->m();

(static_cast<C+>(p2))->k();
```