

Esercizio 3

$P1 \rightarrow k()$

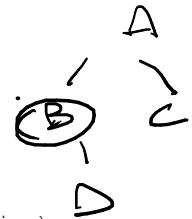
```
class A {
protected:
    virtual void r() {cout<<" A::r ";}
public:
    virtual void g() const {cout <<" A::g ";}
    virtual void f() {cout <<" A::f "; g(); r();}
    void m() {cout <<" A::m "; g(); r();}
    virtual void k() {cout <<" A::k "; r(); m(); }
    virtual A* n() {cout <<" A::n "; return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

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

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

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

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



Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

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

Esercizio 3

p2 → F W

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

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

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

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



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

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

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

Esercizio 3

P2 → ML

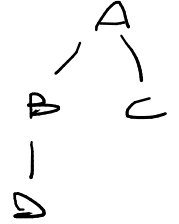
```
class A {
protected:
    virtual void r() {cout<<" A::r ";}
public:
    virtual void g() const {cout <<" A::g ";}
    virtual void f() {cout <<" A::f "; g(); r();}
    void m() {cout <<" A::m "; g(); r();}
    virtual void k() {cout <<" A::k "; r(); m(); }
    virtual A* n() {cout <<" A::n "; return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

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

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

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

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



Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

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

Esercizio 3

P3 → KL

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

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

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

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



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

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

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

Esercizio 3

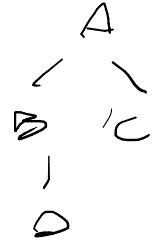
P3 → R0

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

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

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

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



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

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

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

Esercizio 3

p5→g();

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

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

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

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

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

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

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

Esercizio 3

(p3->n())->m(); → RUNS

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

virtualizz
toss

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

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

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



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

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

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

Esercizio 3

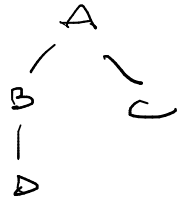
$(p3 \rightarrow n()) \rightarrow n() \rightarrow g();$

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



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

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



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

$A^* p1 = new D(); A^* p2 = new B(); A^* p3 = new C(); B^* p4 = new D(); const A^* p5 = new C();$

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

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


$(p4 \rightarrow n()) \rightarrow m();$

Esercizio 3

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

Restituisce l'istanza

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

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

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



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

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

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

(p5→n())→g();

Esercizio 3

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

NON CONST

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

CONST

NON COMPILA

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

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



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

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

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

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

Esercizio 3

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

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

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

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

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

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

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

Esercizio 3

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

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

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

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

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

ERT = ERROR RUN TIME

loop

*→ C * p2 = new C();*

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

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

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