Esercizi di Programmazione ad Oggetti

Lista n. 7

Esercizio 1

Si consideri la seguente gerarchia di classi.

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

class B {public: virtual ~B() {} };

class C: virtual public A, virtual public B {
public:
  C() {cout << "C0 ";}
  C(B* x, A y = A()) {cout << "C1-2 ";}
};

class D: public C {
public:
  D() {cout << "D0 ";}
  D(C x) {cout << "D1 ";}
};
```

Si considerino le seguenti definizioni di variabili globali ed i seguenti `main()` composti di una singola istruzione. Per ognuno dei `main()` determinare se compila o meno e la stampa che produce in output nel caso invece compili.

```
A a; B b; C c; D d;

main(){D z0(a);}

main(){D z1(b);}

main(){D z2(&b);}

main(){D z3(&c);}

main(){D z4(c);}

main(){D z5(d);}

main(){C z6(&a,c);}

main(){C z7(&c);}

main(){C z8(&d,d);}
```

Esercizio 2

Si consideri la seguente gerarchia di classi e le seguenti variabili globali:

```
class A {public: virtual ~A() {} };
class B: virtual public A {};
class C: virtual public A {};
class D: virtual public A {};
class E: virtual public C {};
class F: virtual public C {};
class G: public B, public E, public F {};

B b; D d; E e; F f; G g; A* pa; B* pb; C* pc; F* pf;
```

Per ognuno dei `main()` determinare se compila o meno e la stampa che produce in output nel caso invece compili.

```
main() {pc = &e; cout << (dynamic_cast<D*> (pc) ? "OK" : "NO");}

main() {cout << (dynamic_cast<B*> (&g) ? "OK" : "NO");}
```

```
main() {pa = &f; cout << (dynamic_cast<C*> (pa) ? "OK" : "NO");}

main() {pb = &b; cout << (dynamic_cast<G*> (pb) ? "OK" : "NO");}

main() {cout << (dynamic_cast<D*> (&d) ? "OK" : "NO");}

main() {pf = &g; cout << (dynamic_cast<E*> (pf) ? "OK" : "NO");}

main() {pf = &f; cout << (dynamic_cast<E*> (pf) ? "OK" : "NO");}
```

Esercizio 3

Si considerino le seguenti definizioni di classe e funzione:

```
class A {
public:
  virtual ~A() {};
};
class B: public A {};
class C: virtual public B {};
class D: virtual public B {};
class E: public C, public D {};

char F(A* p, C& r) {
  B* punt1 = dynamic_cast<B*> (p);
  try{
    E& s = dynamic_cast<E&> (r);
  }
  catch(bad_cast) {
    if(punt1) return 'O';
    else return 'M';
  }
  if(punt1) return 'R';
  return 'A';
}
```

Si consideri inoltre il seguente `main()` incompleto, dove `?` è semplicemente un simbolo per una incognita:

```
main(){
  A a; B b; C c; D d; E e;
  cout << F(?,?) << F(?,?) << F(?,?) << F(?,?);
}
```

Definire opportunamente le chiamate in tale `main()` usando gli oggetti `a`, `b`, `c`, `d`, `e` locali al `main()` in modo tale che la sua esecuzione provochi la stampa `ROMA`.

Esercizio 4

```
class D;

class B {
public:
  virtual D* f() =0;
};

class C {
public:
  virtual C* g();
  virtual B* h() =0;
};

class D: public B, public C {
public:
```

```
  D* f() {cout << "D::f ";  return new D;}
  D* h() {cout << "D::h "; return dynamic_cast<D*>(g());}
};

C* C::g() {
  cout << "C::g ";
  B* p = dynamic_cast<B*>(this);
  if(p) return p->f(); else return this;
}

class E: public D {
public:
  E* f() {
    cout << "E::f ";
    E* p = dynamic_cast<E*>(g());
    if(p) return p; else return this;
  }
};

class F: public E {
public:
  E* g() {cout << "F::g "; return new F;}
  E* h() {
    cout << "F::h ";
    E* p = dynamic_cast<E*>(E::g());
    if(p) return p; else return new F;
  }
};

B* p; C* q; D* r;
```

La compilazione delle precedenti definizioni non provoca errori (con gli opportuni `include` e `using`). Si supponga che ognuno dei seguenti frammenti sia il codice di un `main()` che può accedere alle precedenti definizioni. Si scriva nell'apposito spazio contiguo:

- **NON COMPILA** quando tale `main()` non compila;
- **ERRORE RUN-TIME** quando tale `main()` compila ma l'esecuzione provoca un errore run-time;
- la stampa che produce in output su `cout` nel caso in cui tale `main()` compili ed esegua senza errori; se non provoca alcuna stampa si scriva **NESSUNA STAMPA**.

`p = new E; p->h();`	
`p = new E; p->f();`	
`p = new D; (dynamic_cast<D*>(p))->h();`	
`q = new D; q->g();`	
`q = new E; q->h();`	
`q = new F; q->g();`	
`r = new E; r->f();`	
`r = new F; r->f();`	
`r = new F; r->g();`	
`r = new F; r->h();`	