Esercizi

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

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

B* b = new B(); A* a = new B();
```

```
// COMPILA?
// ERRORE RUN-TIME?
// COSA STAMPA?

b->f();
b->m();
b->k();
a->f();
a->m();
a->k();
(b->n())->g();
(b->n())->n()->g();
(a->n())->g();
(a->n())->m();
```

Definire una superclasse Auto i cui oggetti rappresentano generiche automobili e due sue sottoclassi Benzina e Diesel, i cui oggetti rappresentano automobili alimentate, rispettivamente, a benzina e a diesel (ovviamente non esistono automobili non alimentate e si assume che ogni auto e` o benzina o diesel). Ci interessera` l'aspetto fiscale delle automobili, cioe` il calcolo del bollo auto. Queste classi devono soddisfare le seguenti specifiche:

– Ogni automobile e` caratterizzata dal numero di cavalli fiscali. La tassa per cavallo fiscale e` unica per tutte le automobili (sia benzina che diesel) ed e` fissata in 5 euro. La classe Auto fornisce un metodo tassa() che ritorna la tassa di bollo fiscale per l'automobile di invocazione.
– La classe Diesel e` dotata (almeno) di un costruttore ad un parametro intero x che permette di creare un'auto diesel di x cavalli fiscali. Il calcolo del bollo fiscale per un'auto diesel viene fatto nel seguente modo: si moltiplica il numero di cavalli fiscali per la tassa per cavallo fiscale e si somma una addizionale fiscale unica per ogni automobile diesel fissata in 100 euro.
– Un'auto benzina puo` soddisfare o meno la normativa europea Euro5. La classe Benzina e` dotata di (almeno) un costruttore ad un parametro intero x e ad un parametro booleano b che permette di creare un'auto benzina di x cavalli fiscali che soddisfa Euro5 se b vale true altrimenti che non soddisfa Euro5. Il calcolo del bollo fiscale per un'auto benzina viene fatto nel seguente modo: si moltiplica il numero di cavalli fiscali per la tassa per cavallo fiscale; se l'auto soddisfa Euro5 allora si detrae un bonus fiscale unico per ogni automobile benzina fissato in 50 euro, altrimenti non vi e` alcuna detrazione.

Si definisca inoltre una classe ACI i cui oggetti rappresentano delle filiali ACI addette all'incasso dei bolli auto. Ogni oggetto ACI e` caratterizzato da un array di puntatori ad auto che rappresenta le automobili gestite dalla filiale ACI. La classe ACI fornisce un metodo incassaBolli() che ritorna la somma totale dei bolli che devono pagare tutte le auto gestite dalla filiale di invocazione.

Esercizio 3

Sia `B` una classe polimorfa e sia `C` una sottoclasse di `B`. Definire una funzione `int Fun(const vector<B*>& v)` con il seguente comportamento: sia `v` non vuoto e sia `T*` il tipo dinamico di `v[0]`; allora `Fun(v)` ritorna il numero di elementi di `v` che hanno un tipo dinamico `T1*` tale che `T1` è un sottotipo di `C` diverso da `T`; se `v` è vuoto deve quindi ritornare 0. Ad esempio, il seguente programma deve compilare e provocare le stampe indicate.

```
#include<iostream>
#include<typeinfo>
#include<vector>
using namespace std;

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

int Fun(vector<B*> &v){...}

main() {
  vector<B*> u, v, w;
  cout << Fun(u); // stampa 0
  B b; C c; D d; E e; B *p = &e, *q = &c;
  v.push_back(&c); v.push_back(&b); v.push_back(&d); v.push_back(&c);
  v.push_back(&e); v.push_back(p);
  cout << Fun(v); // stampa 2
  w.push_back(p); w.push_back(&d); w.push_back(q); w.push_back(&e);
  cout << Fun(w); // stampa 1
}
```

Esercizio 11.18

Si consideri il seguente modello di realtà concernente i file audio memorizzati in un riproduttore audio digitale iZod©.

(A) Definire la seguente gerarchia di classi.

1. Definire una classe base polimorfa astratta `FileAudio` i cui oggetti rappresentano un file audio memorizzabile in un iZod. Ogni `FileAudio` è caratterizzato dal titolo (una stringa) e dalla propria dimensione in MB. La classe è astratta in quanto prevede i seguenti **metodi virtuali puri**:
   - un metodo di "clonazione": `FileAudio* clone()`.
   - un metodo `bool qualita()` con il seguente contratto: `f->qualita()` ritorna true se il file audio `*f` è considerato di qualità, altrimenti ritorna false.
2. Definire una classe concreta `Mp3` derivata da `FileAudio` i cui oggetti rappresentano un file audio in formato mp3. Ogni oggetto `Mp3` è caratterizzato dal proprio bitrate espresso in Kbit/s. La classe `Mp3` implementa i metodi virtuali puri di `FileAudio` come segue:
   - per ogni puntatore `p` a `Mp3`, `p->clone()` ritorna un puntatore ad un oggetto `Mp3` che è una copia di `*p`.
   - per ogni puntatore `p` a `Mp3`, `p->qualita()` ritorna true se il bitrate di `*p` è ≥ 192 Kbit/s, altrimenti ritorna false.
3. Definire una classe concreta `WAV` derivata da `FileAudio` i cui oggetti rappresentano un file audio in formato WAV. Ogni oggetto `WAV` è caratterizzato dalla propria frequenza di campionamento espressa in kHz e dall'essere lossless oppure no (cioè con compressione senza perdita oppure con perdita). La classe `WAV` implementa i metodi virtuali puri di `FileAudio` come segue:
   - per ogni puntatore `p` a `WAV`, `p->clone()` ritorna un puntatore ad un oggetto `WAV` che è una copia di `*p`.
   - per ogni puntatore `p` a `WAV`, `p->qualita()` ritorna true se la frequenza di campionamento di `*p` è ≥ 96 kHz, altrimenti ritorna false.

```
class C {
public:
  static void f(const C& x) {}
};

class D {
public:
  D(C x = C()) {}
  void g() const {}
};

class E {
public:
  E(D x = D()) {}
  operator C() const {return C();}
  static void h(const E& x) {C::f(x);}
  void i() const {C::f(*this);}
};

C c; D d; E e;
```

Si supponga che tali definizioni siano visibili ad ognuna delle seguenti istruzioni. Per ognuna di tali istruzioni si scriva nell'apposito spazio contiguo:

- **COMPILA** se la compilazione dell'istruzione non provoca alcun errore;
- **NON COMPILA** se la compilazione dell'istruzione provoca un errore.

`E::h(c);`	
`c.g();`	
`E::h(d);`	
`e.i();`	
`C::f(d);`	
`C::f(e);`	
`d.i();`	
`E e1(c);`	
`D d1(c);`	
`C c1(e);`	
`C c2(d);`	