Esercizio Cosa Stampa

```
class B {
public:
  int x;
  B(int z=1): x(z) {}
  virtual void f() const {cout << x << " B::f() ";}
};

class D: virtual public B {
public:
  virtual void f() const {cout << "D::f() ";}
};

class F: public E, public D {
public:
  F(): B(3) {}
  virtual void f() const {cout << x << " F::f() ";}
  virtual void g() const {cout << "F::g() ";}
};

void Fun(const vector<B*>& v) {
  auto it1 = v.begin();
  vector<B*>::const_iterator it2;
  C* q;
  for(int i=1 ; it1 != v.end(); ++it1, ++i) {
    std::cout << "#" << i << " ";
    (*it1)->f();
    it2 = it1 + 1;
    if(it2 != v.end() && typeid(**it1) == typeid(**it2)) (*it2)->f();
    q = dynamic_cast<C*>(*it1);
    if(q) {static_cast<C*>(q)->g(); q->h();}
    cout << endl;
  }
}

int main() {
  B b; C c; D d; E e; F f;
  vector<B*> v = { &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };
  Fun(v);
}
```

```
class C: virtual public B {
public:
  virtual void g() const {cout << "C::g() ";}
  virtual void h() const {cout << "C::h() ";}
};

class E: public C {
public:
  virtual void f() const {cout << "E::f() ";}
  virtual void h() const {cout << "E::h() ";}
};
```

```
vector<B*> v = { &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };
Fun(v);
```

Le precedenti definizioni compilano correttamente ed il `main` esegue senza undefined behavior o errori run-time. Scrivere nell'apposito spazio relativo alla riga `#i` le stampe prodotte in output dall'iterazione `i`-esima del ciclo for della funzione `fun`, scrivendo **NESSUNA STAMPA** se in una iterazione non ci fossero stampe prodotte in output.

```
#1 ..........
#2 ..........
#3 ..........
#4 ..........
#5 ..........
#6 ..........
#7 ..........
#8 ..........
#9 ..........
#10 ..........
#11 ..........
#12 ..........
```

Esercizio Cosa Stampa

```
class B {
public:
  int x;
  B(int z=1): x(z) {}
  virtual void f() const {cout << x << " B::f() ";}
};

class D: virtual public B {
public:
  virtual void f() const {cout << "D::f() ";}
};


class F: public E, public D {
public:
  F(): B(3) {}
  virtual void f() const {cout << x << " F::f() ";}
  virtual void g() const {cout << "F::g() ";}
};

void Fun(const vector<B*>& v) {
  auto it1 = v.begin();
  vector<B*>::const_iterator it2;
  C* q;
  for(int i=1 ; it1 != v.end(); ++it1, ++i) {
    std::cout << "#" << i << " ";
    (*it1)->f();
    it2 = it1 + 1;
    if(it2 != v.end() && typeid(**it1) == typeid(**it2)) (*it2)->f();
    q = dynamic_cast<C*>(*it1);
    if(q) {static_cast<C*>(q)->g(); q->h();}
    cout << endl;
  }
}

int main() {
  B b; C c; D d; E e; F f;
  vector<B*> v = {  &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };
  Fun(v);
}
```

```
class C: virtual public B {
public:
  virtual void g() const {cout << "C::g() ";}
  virtual void h() const {cout << "C::h() ";}
};


class E: public C {
public:
  virtual void f() const {cout << "E::f() ";}
  virtual void h() const {cout << "E::h() ";}
};
```

```
vector<B*> v = { &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };
```

Le precedenti definizioni compilano correttamente ed il `main` esegue senza undefined behavior o errori run-time. Scrivere nell'apposito spazio relativo alla riga `#i` le stampe prodotte in output dall'iterazione `i`-esima del ciclo for della funzione `fun`, scrivendo **NESSUNA STAMPA** se in una iterazione non ci fossero stampe prodotte in output.

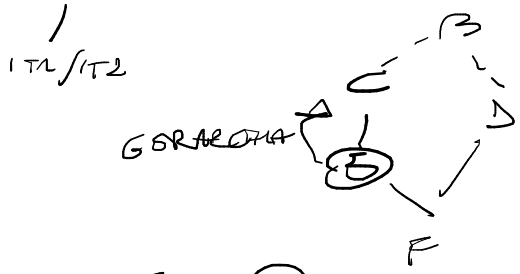
```
#1 ..........

#2 ..........

#3 ..........

#4 ..........

#5 ..........

#6 ..........

#7 ..........

#8 ..........

#9 ..........

#10 ..........

#11 ..........

#12 ..........
```

Esercizio Cosa Stampa

```
class B {
public:
  int x;
  B(int z=1): x(z) {}
  virtual void f() const {cout << x << " B::f() ";}
};

class D: virtual public B {
public:
  virtual void f() const {cout << "D::f() ";}
};

class F: public E, public D {
public:
  F(): B(3) {}
  virtual void f() const {cout << x << " F::f() ";}
  virtual void g() const {cout << "F::g() ";}
};

void Fun(const vector<B*>& v) {
  auto it1 = v.begin();
  vector<B*>::const_iterator it2;
  C* q;
  for(int i=1 ; it1 != v.end(); ++it1, ++i) {
    std::cout << "#" << i << " ";
    (*it1)->f();
    it2 = it1 + 1;
    if(it2 != v.end() && typeid(**it1) == typeid(**it2)) (*it2)->f();
    q = dynamic_cast<C*>(*it1);
    if(q) {static_cast<C*>(q)->g(); q->h();}
    cout << endl;
  }
}

int main() {
  B b; C c; D d; E e; F f;
  vector<B*> v = {  &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };
  Fun(v);
}
```

```
class C: virtual public B {
public:
  virtual void g() const {cout << "C::g() ";}
  virtual void h() const {cout << "C::h() ";}
};

class E: public C {
public:
  virtual void f() const {cout << "E::f() ";}
  virtual void h() const {cout << "E::h() ";}
};
```

```
vector<B*> v = { &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };
```



Le precedenti definizioni compilano correttamente ed il `main` esegue senza undefined behavior o errori run-time. Scrivere nell'apposito spazio relativo alla riga `#i` le stampe prodotte in output dall'iterazione `i`-esima del ciclo for della funzione `fun`, scrivendo **NESSUNA STAMPA** se in una iterazione non ci fossero stampe prodotte in output.

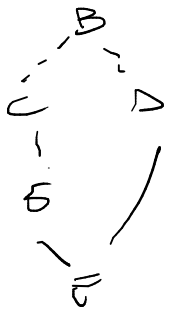
```
#1 ..........................................................................................................

#2 ..........................................................................................................

#3 ..........................................................................................................

#4 ..........................................................................................................

#5 ..........................................................................................................

#6 ..........................................................................................................

#7 ..........................................................................................................

#8 ..........................................................................................................

#9 ..........................................................................................................

#10 .........................................................................................................

#11 .........................................................................................................

#12 .........................................................................................................
```

Esercizio Cosa Stampa

```
class B {
public:
  int x;
  B(int z=1): x(z) {}
  virtual void f() const {cout << x << " B::f() ";}
};

class D: virtual public B {
public:
  virtual void f() const {cout << "D::f() ";}
};

class F: public E, public D {
public:
  F(): B(3) {}
  virtual void f() const {cout << x << " F::f() ";}
  virtual void g() const {cout << "F::g() ";}
};

void Fun(const vector<B*>& v) {
  auto it1 = v.begin();
  vector<B*>::const_iterator it2;
  C* q;
  for(int i=1 ; it1 != v.end(); ++it1, ++i) {
    std::cout << "#" << i << " ";
    (*it1)->f();
    it2 = it1 + 1;
    if(it2 != v.end() && typeid(**it1) == typeid(**it2)) (*it2)->f();
    q = dynamic_cast<C*>(*it1);
    if(q) {static_cast<C*>(q)->g(); q->h();}
    cout << endl;
  }
}

int main() {
  B b; C c; D d; E e; F f;
  vector<B*> v = {  &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };
  Fun(v);
}
```

```
class C: virtual public B {
public:
  virtual void g() const {cout << "C::g() ";}
  virtual void h() const {cout << "C::h() ";}
};

class E: public C {
public:
  virtual void f() const {cout << "E::f() ";}
  virtual void h() const {cout << "E::h() ";}
};
```

```
vector<B*> v = { &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };
```



Le precedenti definizioni compilano correttamente ed il `main` esegue senza undefined behavior o errori run-time. Scrivere nell'apposito spazio relativo alla riga `#i` le stampe prodotte in output dall'iterazione `i`-esima del ciclo for della funzione `fun`, scrivendo **NESSUNA STAMPA** se in una iterazione non ci fossero stampe prodotte in output.

```
#1 ..........

#2 ..........

#3 ..........

#4 ..........

#5 ..........

#6 ..........

#7 ..........

#8 ..........

#9 ..........

#10 ..........

#11 ..........

#12 ..........
```

Esercizio Cosa Stampa

```
class B {
public:
  int x;
  B(int z=1): x(z) {}
  virtual void f() const {cout << x << " B::f() ";}
};

class D: virtual public B {
public:
  virtual void f() const {cout << "D::f() ";}
};

class F: public E, public D {
public:
  F(): B(3) {}
  virtual void f() const {cout << x << " F::f() ";}
  virtual void g() const {cout << "F::g() ";}
};

void Fun(const vector<B*>& v) {
  auto it1 = v.begin();
  vector<B*>::const_iterator it2;
  C* q;
  for(int i=1 ; it1 != v.end(); ++it1, ++i) {
    std::cout << "#" << i << " ";
    (*it1)->f();
    it2 = it1 + 1;
    if(it2 != v.end() && typeid(**it1) == typeid(**it2)) (*it2)->f();
    q = dynamic_cast<C*>(*it1);
    if(q) {static_cast<C*>(q)->g(); q->h();}
    cout << endl;
  }
}

int main() {
  B b; C c; D d; E e; F f;
  vector<B*> v = {  &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };
  Fun(v);
}
```

```
class C: virtual public B {
public:
  virtual void g() const {cout << "C::g() ";}
  virtual void h() const {cout << "C::h() ";}
};

class E: public C {
public:
  virtual void f() const {cout << "E::f() ";}
  virtual void h() const {cout << "E::h() ";}
};
```

vector<B*> v = { &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };

-> B::f (1)

B B

-> B::f (2)

-> NO

Le precedenti definizioni compilano correttamente ed il `main` esegue senza undefined behavior o errori run-time. Scrivere nell'apposito spazio relativo alla riga `#i` le stampe prodotte in output dall'iterazione `i`-esima del ciclo for della funzione `fun`, scrivendo **NESSUNA STAMPA** se in una iterazione non ci fossero stampe prodotte in output.

#1 .....

#2 .....

#3 .....

#4 .....

#5 .....

#6 .....

#7 .....

#8 .....

#9 .....

#10 .....

#11 .....

#12 .....

Esercizio Cosa Stampa

```
class B {
public:
  int x;
  B(int z=1): x(z) {}
  virtual void f() const {cout << x << " B::f() ";}
};

class D: virtual public B {
public:
  virtual void f() const {cout << "D::f() ";}
};

class F: public E, public D {
public:
  F(): B(3) {}
  virtual void f() const {cout << x << " F::f() ";}
  virtual void g() const {cout << "F::g() ";}
};

void Fun(const vector<B*>& v) {
  auto it1 = v.begin();
  vector<B*>::const_iterator it2;
  C* q;
  for(int i=1 ; it1 != v.end(); ++it1, ++i) {
    std::cout << "#" << i << " ";
    (*it1)->f();
    it2 = it1 + 1;
    if(it2 != v.end() && typeid(**it1) == typeid(**it2)) (*it2)->f();
    q = dynamic_cast<C*>(*it1);
    if(q) {static_cast<C*>(q)->g(); q->h();}
    cout << endl;
  }
}

int main() {
  B b; C c; D d; E e; F f;
  vector<B*> v = {  &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };
  Fun(v);
}
```

```
class C: virtual public B {
public:
  virtual void g() const {cout << "C::g() ";}
  virtual void h() const {cout << "C::h() ";}
};

class E: public C {
public:
  virtual void f() const {cout << "E::f() ";}
  virtual void h() const {cout << "E::h() ";}
};
```

vector<B*> v = { &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };

Le precedenti definizioni compilano correttamente ed il `main` esegue senza undefined behavior o errori run-time. Scrivere nell'apposito spazio relativo alla riga `#i` le stampe prodotte in output dall'iterazione `i`-esima del ciclo for della funzione `fun`, scrivendo **NESSUNA STAMPA** se in una iterazione non ci fossero stampe prodotte in output.

```
#1 ..........

#2 ..........

#3 ..........

#4 ..........

#5 ..........

#6 ..........

#7 ..........

#8 ..........

#9 ..........

#10 ..........

#11 ..........

#12 ..........
```

Esercizio Cosa Stampa

```
class B {
public:
  int x;
  B(int z=1): x(z) {}
  virtual void f() const {cout << x << " B::f() ";}
};

class D: virtual public B {
public:
  virtual void f() const {cout << "D::f() ";}
};

class F: public E, public D {
public:
  F(): B(3) {}
  virtual void f() const {cout << x << " F::f() ";}
  virtual void g() const {cout << "F::g() ";}
};

void Fun(const vector<B*>& v) {
  auto it1 = v.begin();
  vector<B*>::const_iterator it2;
  C* q;
  for(int i=1 ; it1 != v.end(); ++it1, ++i) {
    std::cout << "#" << i << " ";
    (*it1)->f();
    it2 = it1 + 1;
    if(it2 != v.end() && typeid(**it1) == typeid(**it2)) (*it2)->f();
    q = dynamic_cast<C*>(*it1);
    if(q) {static_cast<C*>(q)->g(); q->h();}
    cout << endl;
  }
}

int main() {
  B b; C c; D d; E e; F f;
  vector<B*> v = {  &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };
  Fun(v);
}
```

```
class C: virtual public B {
public:
  virtual void g() const {cout << "C::g() ";}
  virtual void h() const {cout << "C::h() ";}
};

class E: public C {
public:
  virtual void f() const {cout << "E::f() ";}
  virtual void h() const {cout << "E::h() ";}
};
```

vector<B*> v = { &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };

Le precedenti definizioni compilano correttamente ed il `main` esegue senza undefined behavior o errori run-time. Scrivere nell'apposito spazio relativo alla riga `#i` le stampe prodotte in output dall'iterazione `i`-esima del ciclo for della funzione `fun`, scrivendo **NESSUNA STAMPA** se in una iterazione non ci fossero stampe prodotte in output.

#1 .....

#2 .....

#3 .....

#4 .....

#5 .....

#6 .....

#7 .....

#8 .....

#9 .....

#10 .....

#11 .....

#12 .....

Esercizio Cosa Stampa

```
class B {
public:
  int x;
  B(int z=1): x(z) {}
  virtual void f() const {cout << x << " B::f() ";}
};

class D: virtual public B {
public:
  virtual void f() const {cout << "D::f() ";}
};

class F: public E, public D {
public:
  F(): B(3) {}
  virtual void f() const {cout << x << " F::f() ";}
  virtual void g() const {cout << "F::g() ";}
};

void Fun(const vector<B*>& v) {
  auto it1 = v.begin();
  vector<B*>::const_iterator it2;
  C* q;
  for(int i=1 ; it1 != v.end(); ++it1, ++i) {
    std::cout << "#" << i << " ";
    (*it1)->f();
    it2 = it1 + 1;
    if(it2 != v.end() && typeid(**it1) == typeid(**it2)) (*it2)->f();
    q = dynamic_cast<C*>(*it1);
    if(q) {static_cast<C*>(q)->g(); q->h();}
    cout << endl;
  }
}

int main() {
  B b; C c; D d; E e; F f;
  vector<B*> v = {  &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };
  Fun(v);
}
```

```
class C: virtual public B {
public:
  virtual void g() const {cout << "C::g() ";}
  virtual void h() const {cout << "C::h() ";}
};

class E: public C {
public:
  virtual void f() const {cout << "E::f() ";}
  virtual void h() const {cout << "E::h() ";}
};
```

vector<B*> v = { &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };

Le precedenti definizioni compilano correttamente ed il `main` esegue senza undefined behavior o errori run-time. Scrivere nell'apposito spazio relativo alla riga `#i` le stampe prodotte in output dall'iterazione `i`-esima del ciclo for della funzione `fun`, scrivendo **NESSUNA STAMPA** se in una iterazione non ci fossero stampe prodotte in output.

```
#1 ..........

#2 ..........

#3 ..........

#4 ..........

#5 ..........

#6 ..........

#7 ..........

#8 ..........

#9 ..........

#10 ..........

#11 ..........

#12 ..........
```

Esercizio Cosa Stampa

```
class B {
public:
  int x;
  B(int z=1): x(z) {}
  virtual void f() const {cout << x << " B::f() ";}
};

class D: virtual public B {
public:
  virtual void f() const {cout << "D::f() ";}
};

class F: public E, public D {
public:
  F(): B(3) {}
  virtual void f() const {cout << x << " F::f() ";}
  virtual void g() const {cout << "F::g() ";}
};

void Fun(const vector<B*>& v) {
  auto it1 = v.begin();
  vector<B*>::const_iterator it2;
  C* q;
  for(int i=1 ; it1 != v.end(); ++it1, ++i) {
    std::cout << "#" << i << " ";
    (*it1)->f();
    it2 = it1 + 1;
    if(it2 != v.end() && typeid(**it1) == typeid(**it2)) (*it2)->f();
    q = dynamic_cast<C*>(*it1);
    if(q) {static_cast<C*>(q)->g(); q->h();}
    cout << endl;
  }
}

int main() {
  B b; C c; D d; E e; F f;
  vector<B*> v = {  &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };
  Fun(v);
}
```

```
class C: virtual public B {
public:
  virtual void g() const {cout << "C::g() ";}
  virtual void h() const {cout << "C::h() ";}
};

class E: public C {
public:
  virtual void f() const {cout << "E::f() ";}
  virtual void h() const {cout << "E::h() ";}
};
```

```
vector<B*> v = { &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };
```

Le precedenti definizioni compilano correttamente ed il `main` esegue senza undefined behavior o errori run-time. Scrivere nell'apposito spazio relativo alla riga `#i` le stampe prodotte in output dall'iterazione `i`-esima del ciclo for della funzione `fun`, scrivendo **NESSUNA STAMPA** se in una iterazione non ci fossero stampe prodotte in output.

```
#1 .....

#2 .....

#3 .....

#4 .....

#5 .....

#6 .....

#7 .....

#8 .....

#9 .....

#10 .....

#11 .....

#12 .....
```