Scrivere un programma consistente di esattamente tre classi `A`, `B` e `C` e della sola funzione `main()` che soddisfi le seguenti condizioni:

1. la classe `A` è definita come:

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

2. le classi `B` e `C` devono essere definite per ereditarietà e non contengono alcun membro
3. la funzione `main()` definisce le tre variabili:

   ```
   A* pa = new A; B* pb = new B; C* pc = new C;
   ```

   e nessuna altra variabile (di alcun tipo)
4. la funzione `main()` può **utilizzare solamente** espressioni di tipo `A*`, `B*` e `C*`, **non può** sollevare eccezioni mediante una `throw` e **non può** invocare l'operatore `new`
5. il programma deve compilare correttamente
6. l'esecuzione di `main()` **deve provocare un errore run-time**.

```
class B: public A {};
class C: public A {};
int main() {/* ...*/ dynamic_cast<C&>(*pb);}
```

Soluzione

Dereferencing a NULL pointer is undefined behavior.

In fact the standard calls this exact situation out in a note (8.3.2/4 "References"):

> Note: in particular, a null reference cannot exist in a well-defined program, because the only way to create such a reference would be to bind it to the "object" obtained by dereferencing a null pointer, which causes undefined behavior.

Ognuno dei seguenti frammenti è il codice di uno o piú metodi pubblici di una qualche classe C. La loro compilazione provoca errori?

`C f(C& x) {return x;}`	OK
`C& g() const {return *this;}`	NC
`C h() const {return *this;}`	OK
`C* m() {return this;}`	OK
`C* n() const {return this;}`	NC
`void p() {}` `void q() const {p();}`	NC
`void p() {}` `static void r(C *const x) {x->p();}`	OK
`void s(C *const x) const {*this = *x;}`	NC
`static C& t() {return C();}`	NC
`static C *const u(C& x) {return &x;}`	OK

```
class Z {
 public: Z(int x) {}
};
```

```
class A {
 public:
  void f(int) {cout << "A::f(int) "; f(true);}
  virtual void f(bool) {cout <<"A::f(bool) ";}
  virtual A* f(Z) {cout <<"A::f(Z) "; f(2); return this;}
  A() {cout <<"A() "; }
};
```

```
class B: virtual public A {
 public:
  void f(const bool&){cout<< "B::f(const bool&) ";}
  void f(const int&){cout<< "B::f(const int&) ";}
  virtual B* f(Z) {cout <<"B::f(Z) "; return this;}
  virtual ~B() {cout << "~B ";}
  B() {cout <<"B() "; }
};
```

```
class C: virtual public A {
 public:
  C* f(Z){cout <<"C::f(Z) "; return this;}
  C() {cout <<"C() "; }
};
```

```
class D: virtual public A {
public:
  virtual void f(bool) const {cout <<"D::f(bool) ";}
  A* f(Z) {cout << "D::f(Z) "; return this;}
  ~D() {cout <<"~D ";}
  D() {cout <<"D() ";}
};
```

```
class E: public C {
 public:
  C* f(Z){cout <<"E::f(Z) "; return this;}
  ~E() {cout <<"~E ";}
  E() {cout <<"E() ";}
};
```

```
class F: public B, public E, public D  {
 public:
  void f(bool){cout<< "F::f(bool) ";}
  F* f(Z){cout <<"F::f(Z) "; return this;}
  F() {cout <<"F() "; }
  ~F() {cout <<"~F ";}
};
```

```
B* pb=new B; C* pc = new C; D* pd = new D; E* pe = new E;
F* pf = new F; B *pb1= new F;
A *pa1=pb, *pa2=pc, *pa3=pd, *pa4=pe, *pa5=pf;
```

cosa stampa?

```
E* puntE = new F;                          ........
A* puntA = new F;                          ........
pa3->f(3);                                 ........
pa5->f(3);                                 ........
pb1->f(true);                              ........
pa4->f(true);                              ........
pa2->f(Z(2));                              ........
pa5->f(Z(2));                              ........
(dynamic_cast<E*>(pa4))->f(Z(2));          ........
(dynamic_cast<C*>(pa5))->f(Z(2));          ........
pb->f(3);                                  ........
pc->f(3);                                  ........
(pa4->f(Z(3)))->f(4);                      ........
(pc->f(Z(3)))->f(4);                       ........
delete pa5;                                ........
delete pb1;                                ........
```