Si considerino le seguenti definizioni.

```
class B {                       class D: public B {
private:                        private:
  vector<bool>* ptr;              int x;
  virtual void m() =0;          };
};

class E: public D {
private:
  list<int*> l;
  int& ref;
  double* p;
public:
  void m() {}
  // ridefinizione del costruttore di copia di E
};
```

Ridefinire il costruttore di copia della classe `E` in modo tale che il suo comportamento coincida con quello del costruttore di copia standard di `E`.

```
E(const E& e) : D(e), l(e.l), ref(e.ref), p(e.p) {}
```

```
class A {                                  class B {
  bool x;                                    bool y;
public:                                    public:
  virtual ~A() {};                           virtual void f() const {cout << "B::f ";}
};                                         };

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

template<class T>
void Fun(const T& ref) {
  const B* pB = dynamic_cast<const B*>(&ref);
  const E* pE = dynamic_cast<const E*>(&ref);
  if(dynamic_cast<const C*>(&ref)) {cout << "C "; ref.f(); return;}
  if(pE) {cout << "E "; pE->f(); return;}
  if(pB) {cout << "B "; pB->f(); return;}
  if(dynamic_cast<const A*>(&ref)) {cout << "A "; return;}
  if(dynamic_cast<const D*>(&ref)) {cout << "D "; return;}
}

C c; D d; E e;
A& a1 = c; B& b1 = d; B& b2 = e; B* b3 = new B(); D& d1 = e;
D* pD1 = dynamic_cast<E*>(&b2); D* pD2 = dynamic_cast<D*>(&b2);
```

Si considerino le precedenti definizioni. Per ognuna delle seguenti istruzioni di invocazione del template di funzione `Fun` scrivere nell'apposito spazio **UNA RISPOSTA PER OGNI LINEA DA 01 A 10 ANCHE IN CASO DI NESSUNA RISPOSTA**:

- **NC** se l'istruzione **N**on **C**ompila correttamente;
- **ERT** se l'istruzione compila correttamente ma la sua esecuzione provoca un **E**rrore a **R**un **T**ime;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **precisamente** la stampa che l'esecuzione produce in output su `cout`; se provoca **N**essuna **S**tampa allora si scriva **NS**.
- **??** in caso di **nessuna risposta**.

```
01:  Fun<A>(c);
02:  Fun(a1);
03:  Fun(b1);
04:  Fun(d1);
05:  Fun(*pD2);
06:  Fun<E>(d1);
07:  Fun<E>(e);
08:  Fun(*b3);
09:  Fun<D>(e);
10:  Fun(b2);
```

```
class C {
public:
  C() {cout << "C0 ";}
  C(const C&) {cout << "Cc ";}
  C& operator=(const C& e) {
    cout << "C= ";
    return *this;
  }
};

class E: public C {
public:
  C c;
  E() {cout << "E0 ";}
  E& operator=(const E& e) {
    *this = e;
    cout << "E= "; return *this;
  }
};

C x1, x2; E y1, y2; F z1, z2;
```

```
class D {
public:
  C c;
  D() {cout << "D0 ";}
D(const D&) {cout <<"Dc ";}
};

class F: public C {
public:
  C* pc;
  F() {cout << "F0 ";}
  F(const F&) {cout <<"Fc ";}
  F& operator=(const F& f) {
    C::operator=(f); pc=f.pc;
    cout << "E= "; return *this;
  }
};
```

Si considerino le precedenti definizioni che compilano senza errori. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio **UNA RISPOSTA PER OGNI LINEA DA 01 A 10 ANCHE IN CASO DI NESSUNA RISPOSTA**:

- **NC** se l'istruzione **N**on **C**ompila correttamente;
- **ERT** se l'istruzione compila correttamente ma la sua esecuzione provoca un **E**rrore a **R**un **T**ime;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva **precisamente** la stampa che l'esecuzione produce in output su `cout`; se provoca **N**essuna **S**tampa allora si scriva **NS**.
- **??** in caso di **nessuna risposta**.

```
01:  D d1;
02:  D d2(d1);
03:  E e1;
04:  E e2=y1;
05:  y1=y2;
06:  F* pf = new F();
07:  F f = z1;
08:  z1=z2;
09:  x1=y1;
10:  z1=x1;
```