Quesito 2

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

class B: virtual public A {};

class C: virtual public A {};

class D: public B, public C {};

class E: public D {};

class F: public E {};

template<class T>
void Fun(T& ref){
  bool b=0;
  B* p = &ref;
  try{ throw ref; }
  catch(E){cout << "E "; b=1;}
  catch(D){cout << "D "; b=1;}
  catch(B){cout << "B "; b=1;}
  catch(A){cout << "A "; b=1;}
  catch(C){cout << "C "; b=1;}
  if(b==0) cout << "ZERO ";
}

A a; B b; C c; D d; E e; F f;
A* pa = &b; D* pd = &f;
B* pb=dynamic_cast<B*>(pa); C* pc=dynamic_cast<E*>(pd); E* pe=static_cast<E*>(pd);
```

Le precedenti definizioni compilano correttamente (con gli opportuni `#include` e `using`). Per ognuna delle seguenti istruzioni di invocazione della funzione `Fun` scrivere nell'apposito spazio:

- **NON COMPILA** se la compilazione dell'istruzione provoca un errore;
- **ERRORE RUN-TIME** se l'istruzione compila correttamente ma la sua esecuzione provoca un errore a run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva la stampa che l'esecuzione produce in output su `cout`; se non provoca alcuna stampa allora si scriva **NESSUNA STAMPA**.

`Fun(a);`	
`Fun(b);`	
`Fun(c);`	
`Fun(d);`	
`Fun(e);`	
`Fun(f);`	
`Fun(*pa);`	
`Fun(*pb);`	
`Fun(*pc);`	
`Fun(*pd);`	
`Fun(*pe);`	
`Fun<B>(*pd);`	
`Fun<D>(*pd);`	
`Fun<E>(*pd);`	