```cpp
#include <iostream>

template<typename T> void p(T x) { std::cout << x; }

class A { };
class B : public A { };
class C : public A { };

void foo() {
    p("1");
    throw C();
}

int main() {
    try {
        p("2");
        foo();
        p("3");
    } catch(B) {
        p("4");
    } catch(C) {
        p("5");
    } catch(A) {
        p("6");
    }
    p("7");
}
```

What might happen if you try to compile this code?

```cpp
#include <iostream>

template<typename T> void p(T x) { std::cout << x; }

class A {
protected:
    void foo() { p("foo"); };
};

class B : virtual public A {
public:
    void bar() { p("bar"); };
};

int main() {
    B b;
    b.foo();
}
```

what might happen if you try to compile, link and run this program?

```
1  #include <iostream>
2
3  template<typename T> void p(T x) { std::cout << x; }
4
5  class A {
6  public:
7      virtual void boo() { p("A"); };
8  };
9
10 class B : public A {
11 public:
12     virtual void boo() { p("B"); };
13 };
14
15 int main() {
16     try {
17         throw B();
18     } catch( A a ) {
19         a.boo();
20     }
21 }
```

what might happen if you try to compile, link and run this program?

```cpp
#include <iostream>

template<typename T> void p(T x) { std::cout << x; }

class A { };

int main() {
    p("0");
    try {
        p("1");
        try {
            p("2");
            throw A();
            p("3");
        } catch(...) {
            p("4");
            throw;
            p("5");
        }
    } catch(A a) {
        p("6");
        throw;
        p("7");
    }
    p("8");
}
```

what might happen if you try to compile, link and run this program?

```
1  #include <iostream>
2
3  template<typename T> void p(T x) { std::cout << x; }
4
5  class B {
6  public:
7      B() { p("B"); }
8      ~B() { p("b"); }
9  };
10
11 class A {
12     B * v;
13 public:
14     A() { p("A"); v = new B[3]; throw "x"; }
15     ~A() { p("a"); delete[] v; }
16 };
17
18 int main() {
19     try {
20         p("0");
21         A a;
22         p("1");
23     } catch( const char * s ) {
24         p(s);
25     }
26 }
```

what might happen if you try to compile, link and run this program?

```
1  #include <iostream>
2
3  template<typename T> void p(T x) { std::cout << x; }
4
5  class B {
6  public:
7      B() { p("B"); throw "x"; }
8      ~B() { p("b"); }
9  };
10
11 class A {
12     B b;
13 public:
14     A() try : b() { p("A"); } catch(...) { p("y"); }
15     ~A() { p("a"); }
16 };
17
18 int main() {
19     try {
20         p("0");
21         A a;
22         p("1");
23     } catch( const char * s ) {
24         p(s);
25     }
26 }
```

what might happen if you try to compile, link and run this program?

```
1   #include <iostream>
2
3   template<typename T> void p(T x) { std::cout << x; }
4
5   void foo() throw() {
6       throw "x";
7   };
8
9   int main() {
10      try {
11          p("0");
12          foo();
13          p("1");
14      } catch( const char * s ) {
15          p(s);
16      }
17  }
```

what might happen if you try to compile, link and run this program?

```
1  #include <iostream>
2
3  template<typename T> void p(T x) { std::cout << x; }
4
5  struct B {
6      void foo(int) { p(1); }
7  };
8
9  struct C : B {
10      void foo(float) { p(2); }
11  };
12
13  int main() {
14      C c;
15      c.foo(42);
16  }
```

what might happen if you try to compile, link and run this program?

```
1  #include <iostream>
2
3  template<typename T> void p(T x) { std::cout << x; }
4
5  struct A {
6      A() { p('A'); }
7  };
8
9  struct B : A {
10     B() { p('B'); }
11 };
12
13 struct C : virtual A {
14     C() { p('C'); }
15 };
16
17 struct D : B, C {
18     D() { p('D'); }
19 };
20
21 int main() {
22     D d;
23 }
```

what might happen if you try to compile, link and run this program?

```cpp
#include <iostream>
#include <typeinfo>

template<typename T> void p(T x) { std::cout << x; }

struct A {
    virtual void foo() { p("A"); }
};

struct B : A {
    virtual void foo() { p("B"); }
};

struct C : A {
    virtual void foo() { p("C"); }
};

int main() {
    A * b = new B;
    A * c = new C;
    b = c;
    b->foo();
    b = dynamic_cast<B*>(c);
    b->foo();
}
```

what might happen if you try to compile, link and run this program?

```
 1  #include <iostream>
 2  #include <typeinfo>
 3
 4  struct Foo {
 5  };
 6
 7  struct Bar : Foo {
 8  };
 9
10  struct Gaz : Foo {
11  };
12
13  int main() {
14      Foo * b = new Bar;
15      Foo * c = new Gaz;
16      std::cout << typeid(*b).name();
17      std::cout << typeid(*c).name();
18  }
```

what might happen if you try to compile, link and run this program?