```
 1 #include <iostream>
 2
 3 class Rectangle {
 4 public:
 5     Rectangle(int width, int height) : width_(width), height_(height) {}
 6     int width() { return width_; }
 7     int height() { return height_; }
 8     int area() { return width() * height(); }
 9 private:
10     int width_;
11     int height_;
12 };
13
14 class Square : public Rectangle {
15 public:
16     Square(int length) : Rectangle(length,length) {}
17 };
18
19 int main() {
20     Square s(4);
21     std::cout << s.area() << std::endl;
22 }
```

When will be printed when running this code? Please critizise the design.

```
 1 #include <iostream>
 2
 3 struct Foo {
 4     void f() const { std::cout << "Foo" << std::endl; }
 5 };
 6
 7 struct Bar {
 8     void f() const { std::cout << "Bar" << std::endl; };
 9 };
10
11 template <typename T> void f(const T & t) {
12     t.f();
13 }
14
15 int main() {
16     f(Foo());
17     f(Bar());
18 }
```

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 Foo {
 6 public:
 7     void do_it() { p(1); func(); }
 8 private:
 9     virtual void func() { p(2); }
10 };
11
12 class Bar : public Foo {
13 public:
14     void do_it() { p(3); func(); }
15 private:
16     void func() { p(4); }
17 };
18
19 void run(Foo & f) {
20     f.do_it();
21 }
22
23 int main() {
24     Foo f;
25     Bar b;
26     run(f);
27     run(b);
28 }
```

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

```
1 #include "Foo.hpp"
2 #include "Bar.hpp"
3
4 class Gaz {
5     Foo * f;
6     Bar & b;
7 public:
8     // ...
9 };
```

Please critizise this code.

```
 1 #include <iostream>
 2
 3 // a silly implementation of a vector of integers
 4 class IntVector {
 5     int v[1024];
 6 public:
 7     int size() { return sizeof(v) / sizeof(v[0]); }
 8     int & operator[](int i) { return v[i]; }
 9 };
10
11 // a silly attempt to make a safe version of IntVector
12 class SafeIntVector : public IntVector {
13 public:
14     class OutOfRange {};
15     int & operator[](int i) {
16         if (i < 0 || i >= size())
17             throw OutOfRange();
18         return IntVector::operator[](i);
19     }
20 };
21
22 int main() {
23     SafeIntVector v;
24     v[4] = 42;
25     std::cout << v[4] << std::endl;
26 }
```

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

```
 1 #include <iostream>
 2
 3 class Action {
 4 public:
 5     virtual void do_it() const = 0;
 6     virtual ~Action() { }
 7 };
 8
 9 class SayGreeting : public Action {
10     std::ostream & ostm_;
11 public:
12     SayGreeting() : ostm_(std::cout) {}
13     void do_it() const { ostm_ << "Hello" << std::endl; }
14 };
15
16 class SayGoodbye : public Action {
17     std::ostream & ostm_;
18 public:
19     SayGoodbye() : ostm_(std::cout) {}
20     void do_it() const { ostm_ << "Good Bye!" << std::endl; }
21 };
22
23 void execute(const Action & action) {
24     action.do_it();
25 }
26
27 int main() {
28     execute(SayGreeting());
29     execute(SayGoodbye());
30 }
```

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 template <int min, int max> class IntRange {
 6     int value_;
 7 public:
 8     class Error {};
 9     IntRange(int value) : value_(value) {
10         if (value_ < min || value_ > max)
11             throw Error();
12     }
13     IntRange operator=(int i) {
14         return *this = IntRange(i);
15     }
16     operator int() {
17         return value_;
18     }
19     // ...
20 };
21
22 int main() {
23     IntRange<1,12> r(3);
24     p(1);  r = 1;
25     p(2);  r = 12;
26     p(3);  r = 13;
27     p(4);  r = 14;
28 }
```

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