Casts in C++
Type casting in C++

- C syntax casts: `int* x = (double*) y;`
- `dynamic_cast`
- `static_cast`
- `reinterpret_cast`
- `const_cast`
can perform conversions between pointers to related classes
performs downcast as well
No checks are performed during runtime

```cpp
class Base {}
class Derived: public Base {}
Base * a = new Base;
Derived * b = static_cast<Derived*>(a);
```
• can only be used with pointers and references to classes
• can be used for upcast and downcast
• downcast available only for polymorphic classes
• checks that pointed object is a valid complete object of the target type
• run time overhead
• used to cast from virtual bases
class Base { virtual void dummy() {});
class Derived: public Base {};
int main () {
try {
    Base * pba = new Derived;
    Base * pbb = new Base;
    Derived * pd;
    pd = dynamic_cast<Derived*>(pba);
    if (pd==0) cout << "Null pointer on first type-cast."
    pd = dynamic_cast<Derived*>(pbb);
    if (pd==0) cout << "Null pointer on second type-cast."
}
catch (exception& e) { cout << "Exception: " << e.what();
return 0; 
}
reinterpret_cast

• converts any pointer type to any other pointer type, even of unrelated classes
• neither the content pointed nor the pointer type itself is checked
• mostly used for low-level operations
struct A {};
struct B : A { virtual void f() {} };
struct C : B {}

A* a = new B();
B* b1 = static_cast<B*>(a);
B* b2 = reinterpret_cast<B*>(a);
B* b3 = dynamic_cast<B*>(a);

B* b = new C();
C* c1 = static_cast<C*>(b);
C* c2 = reinterpret_cast<C*>(b);
C* c3 = dynamic_cast<C*>(b);

What is the difference?
casting and virtual inheritance

struct A { virtual void f(){}; };
struct B : virtual A { };
A* a = new B();
B* b1 = static_cast<B*>(a); // Compiler error
B* b2 = reinterpret_cast<B*>(a); // Runs, but a wrong result
B* b3 = dynamic_cast<B*>(a);

why?