```
#include <iostream>
#include <string>
using namespace std;
class Base
{
   public:
      int f()const;
      int f(string) const;
      void g();
};
int Base :: f()const
{
   cout << "Base::f()\n";
   return 1;
}
int Base :: f(string s)const
{
   cout << s << endl;
   return 1;
}
void Base :: g()
{

}
class Derived1 : public Base
{
   public:
      void g() const;
};
void Derived1 :: g()const
{

}
class Derived2 : public Base
{
   public:
      int f() const;              // Redefinition:
};
int Derived2 :: f() const
{
   cout << "Derived2::f()\n";
   return 2;
}
class Derived3 : public Base
{
   public:
      void f() const;             // Change return type:
};
```

```
void Derived3 :: f() const
{
   cout << "Derived3::f()\n";
}
class Derived4 : public Base
{
   public:// Change argument list:
      int f(int) const;               // Change argument list:
};
int Derived4 :: f(int) const
{
   cout << "Derived4::f()\n";
   return 4;
}
int main()
{
   string s("hello");
   Derived1 d1;

   int x = d1.f();
   d1.f(s);
   cout << "x = " << x << endl;

   Derived2 d2;
   x = d2.f();        //!  d2.f(s); // string version hidden
   cout << "x = " << x << endl;

   Derived3 d3;


   Derived4 d4;       //!  x = d4.f(); // f() version hidden

   x = d4.f(1);
   cout << "x = " << x << endl;
   return 0;
}
```