```
#include<iostream>
#include<string>
using namespace std;

class Z {
public:
  operator int() const {return 0;}
};

template<class T> class D; // dichiarazione incompleta

template<class T1, class T2 = Z, int k = 1>
class C {
  friend class D<T1>;
private:
  T1 t1;
  T2 t2;
  int a;
  C(int x =k): a(x) {}
};

template<class T>
class D {
 public:
  void f() const {C<T,T> c(1); cout << c.t1 << c.t2 << c.a;}
  void g() const {C<int> c;}
  void h() const {C<T,int> c(3); cout << c.t2 << c.a;}
  void m() const {C<int,T,3> c; cout << c.t1;}
  void n() const {C<int,double> c; cout << c.t1 << c.t2 << c.a;}
  void o() const {C<char,double> c(6); cout << c.a;}
  void p() const {C<Z,T,7> c(7); cout << c.t2 << c.a;}
};
```

Determinare se i seguenti `main()` compilano correttamente o meno barrando la corrispondente scritta.

```
int main() { D<char> d1; d1.f(); }
```

```
int main() { D<std::string> d2; d2.f(); }
```

```
int main() { D<char> d3; d3.g(); }
```

```
int main() { D<int> d4; d4.g(); }
```

```
int main() { D<char> d5; d5.h(); }
```

```
int main() { D<int> d6; d6.h(); }
```

```
int main() { D<char> d7; d7.m(); }
```

```
int main() { D<int> d8; d8.m(); }
```

```
int main() { D<char> d9; d9.n(); }
```

```
int main() { D<Z> d10; d10.n(); }
```

```
int main() { D<char> d11; d11.o(); }
```

```
int main() { D<Z> d12; d12.o(); }
```

```
int main() { D<char> d13; d13.p(); }
```

```
int main() { D<Z> d14; d14.p(); }
```

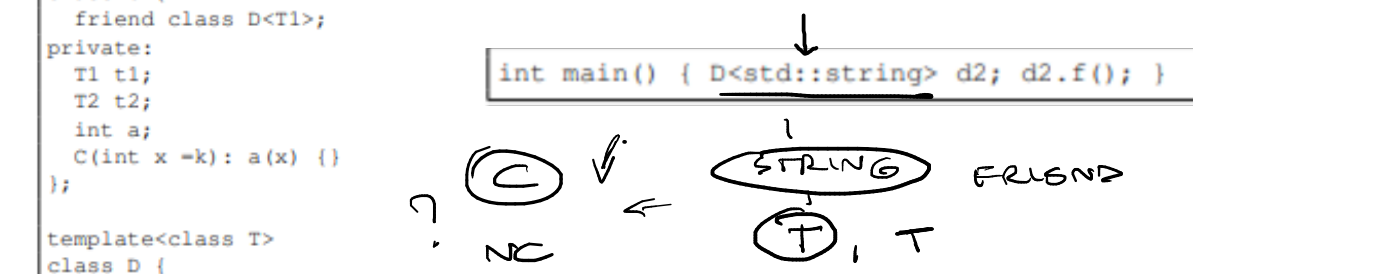
```
// dichiarazione incompleta
template<class T> class D;

template<class T1, class T2>
class C {
  // amicizia associata
  friend class D<T1>;
private:
  T1 t1; T2 t2;
};

template<class T>
class D {
public:
  void m(){C<T,T> c;
           cout << c.t1 << c.t2;}
  void n(){C<int,T> c;
           cout << c.t1 << c.t2;}
  void o(){C<T,int> c;
           cout << c.t1 << c.t2;}
  void p(){C<int,int> c;
           cout << c.t1 << c.t2;}
  void q(){C<int,double> c;
           cout << c.t1 << c.t2;}
  void r(){C<char,double> c;
           cout << c.t1 << c.t2;}
};
```

I seguenti main() compilano?

```
main() {D<char> d; d.m();} // C
main() {D<char> d; d.n();} // NC
main() {D<char> d; d.o();} // C
main() {D<char> d; d.p();} // NC
main() {D<char> d; d.q();} // NC
main() {D<char> d; d.r();} // C
```

```
#include<iostream>
#include<string>
using namespace std;

class Z {
public:
  operator int() const {return 0;}
};

template<class T> class D; // dichiarazione incompleta

template<class T1, class T2 = Z, int k = 1>
class C {
  friend class D<T1>;
private:
  T1 t1;
  T2 t2;
  int a;
  C(int x =k): a(x) {}
};

template<class T>
class D {
 public:
  void f() const {C<T,T> c(1); cout << c.t1 << c.t2 << c.a;}
  void g() const {C<int> c;}
  void h() const {C<T,int> c(3); cout << c.t2 << c.a;}
  void m() const {C<int,T,3> c; cout << c.t1;}
  void n() const {C<int,double> c; cout << c.t1 << c.t2 << c.a;}
  void o() const {C<char,double> c(6); cout << c.a;}
  void p() const {C<Z,T,7> c(7); cout << c.t2 << c.a;}
};
```

```
int main() { D<std::string> d2; d2.f(); }
```

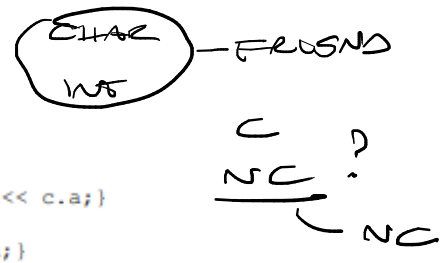


Determinare se i seguenti `main()` compilano correttamente o meno barrando la corrispondente scritta.

```
int main() { D<char> d1; d1.f(); }
int main() { D<std::string> d2; d2.f(); }
int main() { D<char> d3; d3.g(); }
int main() { D<int> d4; d4.g(); }
int main() { D<char> d5; d5.h(); }
int main() { D<int> d6; d6.h(); }
int main() { D<char> d7; d7.m(); }
int main() { D<int> d8; d8.m(); }
int main() { D<char> d9; d9.n(); }
int main() { D<Z> d10; d10.n(); }
int main() { D<char> d11; d11.o(); }
int main() { D<Z> d12; d12.o(); }
int main() { D<char> d13; d13.p(); }
int main() { D<Z> d14; d14.p(); }
```

```
#include<iostream>
#include<string>
using namespace std;

class Z {
public:
  operator int() const {return 0;}
};

template<class T> class D; // dichiarazione incompleta

template<class T1, class T2 = Z, int k = 1>
class C {
  friend class D<T1>;
private:
  T1 t1;
  T2 t2;
  int a;
  C(int x =k): a(x) {}
};

template<class T>
class D {
 public:
  void f() const {C<T,T> c(1); cout << c.t1 << c.t2 << c.a;}
  void g() const {C<int> c;}
  void h() const {C<T,int> c(3); cout << c.t2 << c.a;}
  void m() const {C<int,T,3> c; cout << c.t1;}
  void n() const {C<int,double> c; cout << c.t1 << c.t2 << c.a;}
  void o() const {C<char,double> c(6); cout << c.a;}
  void p() const {C<Z,T,7> c(7); cout << c.t2 << c.a;}
};
```

```
int main() { D<char> d3; d3.g(); }
```



Determinare se i seguenti `main()` compilano correttamente o meno barrando la corrispondente scritta.

```
int main() { D<char> d1; d1.f(); }
```

```
int main() { D<std::string> d2; d2.f(); }
```

```
int main() { D<char> d3; d3.g(); }
```

```
int main() { D<int> d4; d4.g(); }
```

```
int main() { D<char> d5; d5.h(); }
```

```
int main() { D<int> d6; d6.h(); }
```

```
int main() { D<char> d7; d7.m(); }
```

```
int main() { D<int> d8; d8.m(); }
```

```
int main() { D<char> d9; d9.n(); }
```

```
int main() { D<Z> d10; d10.n(); }
```

```
int main() { D<char> d11; d11.o(); }
```

```
int main() { D<Z> d12; d12.o(); }
```

```
int main() { D<char> d13; d13.p(); }
```

```
int main() { D<Z> d14; d14.p(); }
```

```
#include<iostream>
#include<string>
using namespace std;

class Z {
public:
  operator int() const {return 0;}
};

template<class T> class D; // dichiarazione incompleta

template<class T1, class T2 = Z, int k = 1>
class C {
  friend class D<T1>;
private:
  T1 t1;
  T2 t2;
  int a;
  C(int x =k): a(x) {}
};

template<class T>
class D {
 public:
  void f() const {C<T,T> c(1); cout << c.t1 << c.t2 << c.a;}
  void g() const {C<int> c;}
  void h() const {C<T,int> c(3); cout << c.t2 << c.a;}
  void m() const {C<int,T,3> c; cout << c.t1;}
  void n() const {C<int,double> c; cout << c.t1 << c.t2 << c.a;}
  void o() const {C<char,double> c(6); cout << c.a;}
  void p() const {C<Z,T,7> c(7); cout << c.t2 << c.a;}
};
```

int main() { D<char> d5; d5.h(); }

T, WT → COMPILA

Determinare se i seguenti `main()` compilano correttamente o meno barrando la corrispondente scritta.

```
int main() { D<char> d1; d1.f(); }
int main() { D<std::string> d2; d2.f(); }
int main() { D<char> d3; d3.g(); }
int main() { D<int> d4; d4.g(); }
int main() { D<char> d5; d5.h(); }
int main() { D<int> d6; d6.h(); }
int main() { D<char> d7; d7.m(); }
int main() { D<int> d8; d8.m(); }
int main() { D<char> d9; d9.n(); }
int main() { D<Z> d10; d10.n(); }
int main() { D<char> d11; d11.o(); }
int main() { D<Z> d12; d12.o(); }
int main() { D<char> d13; d13.p(); }
int main() { D<Z> d14; d14.p(); }
```

```
#include<iostream>
#include<string>
using namespace std;

class Z {
public:
  operator int() const {return 0;}
};

template<class T> class D; // dichiarazione incompleta

template<class T1, class T2 = Z, int k = 1>
class C {
  friend class D<T1>;
private:
  T1 t1;
  T2 t2;
  int a;
  C(int x =k): a(x) {}
};

template<class T>
class D {
 public:
  void f() const {C<T,T> c(1); cout << c.t1 << c.t2 << c.a;}
  void g() const {C<int> c;}
  void h() const {C<T,int> c(3); cout << c.t2 << c.a;}
  void m() const {C<int,T,3> c; cout << c.t1;}
  void n() const {C<int,double> c; cout << c.t1 << c.t2 << c.a;}
  void o() const {C<char,double> c(6); cout << c.a;}
  void p() const {C<Z,T,7> c(7); cout << c.t2 << c.a;}
};
```

int main() { D<int> d6; d6.h(); }

FRIEND = USO I CAMPI PRIVATI

Determinare se i seguenti `main()` compilano correttamente o meno barrando la corrispondente scritta.

```
int main() { D<char> d1; d1.f(); }
int main() { D<std::string> d2; d2.f(); }
int main() { D<char> d3; d3.g(); }
int main() { D<int> d4; d4.g(); }
int main() { D<char> d5; d5.h(); }
int main() { D<int> d6; d6.h(); }
int main() { D<char> d7; d7.m(); }
int main() { D<int> d8; d8.m(); }
int main() { D<char> d9; d9.n(); }
int main() { D<Z> d10; d10.n(); }
int main() { D<char> d11; d11.o(); }
int main() { D<Z> d12; d12.o(); }
int main() { D<char> d13; d13.p(); }
int main() { D<Z> d14; d14.p(); }
```

```
#include<iostream>
#include<string>
using namespace std;

class Z {
public:
  operator int() const {return 0;}
};

template<class T> class D; // dichiarazione incompleta

template<class T1, class T2 = Z, int k = 1>
class C {
  friend class D<T1>;
private:
  T1 t1;
  T2 t2;
  int a;
  C(int x =k): a(x) {}
};

template<class T>
class D {
 public:
  void f() const {C<T,T> c(1); cout << c.t1 << c.t2 << c.a;}
  void g() const {C<int> c;}
  void h() const {C<T,int> c(3); cout << c.t2 << c.a;}
  void m() const {C<int,T,3> c; cout << c.t1;}
  void n() const {C<int,double> c; cout << c.t1 << c.t2 << c.a;}
  void o() const {C<char,double> c(6); cout << c.a;}
  void p() const {C<Z,T,7> c(7); cout << c.t2 << c.a;}
};
```

int main() { D<char> d13; d13.p(); }

CHAR != INT

Determinare se i seguenti `main()` compilano correttamente o meno barrando la corrispondente scritta.

```
int main() { D<char> d1; d1.f(); }
int main() { D<std::string> d2; d2.f(); }
int main() { D<char> d3; d3.g(); }
int main() { D<int> d4; d4.g(); }
int main() { D<char> d5; d5.h(); }
int main() { D<int> d6; d6.h(); }
int main() { D<char> d7; d7.m(); }
int main() { D<int> d8; d8.m(); }
int main() { D<char> d9; d9.n(); }
int main() { D<Z> d10; d10.n(); }
int main() { D<char> d11; d11.o(); }
int main() { D<Z> d12; d12.o(); }
int main() { D<char> d13; d13.p(); }
int main() { D<Z> d14; d14.p(); }
```