1. 在数据库的查询操作中，对 SQL 语句的查询构造过程一般采用如下类似的方式：

```
int main( )
{
  CMyString userName, password;
  cin >> userName >> password;
  CMyString sql = "select * from USER where userName="+userName+"and password ="+password;
  return 0;
}
```

这种方法易出现的问题是，

如果 1：useName="user"，password = "  || 1 == 1"，那么 where 字句中的密码查询结果都是 true，造成密码失效。

如果 2：useName="user --"，password = "abc"，那么 where 字句中“user”后面的 ”-- “ 使得 password 语句成为注释，也会造成密码失效。

为了防止出现类似 SQL 语句的注入错误，在 ODBC 或 JDBC 等的数据库连接与查询过程中，SQL 语句考虑采用以下方法来构造：

```
int main( )
{
  CSqlStatement sql ="select ?, ? from student where user = ?  and password = ?";
  sql.SetAttribute("1", "SID");
  sql.SetAttribute("2", "Name");
  sql.SetAttribute("3", "BIT");
  sql.SetAttribute("4", "CPP");
  sql.ExecuteSql( );          //  输出是：select SID, Name from student where user = BIT and password = CPP

  sql ="select ?, ? from student where user = ?  and password = ?";
  sql.SetAttribute("1", "SID");
  sql.SetAttribute("2", "Name");
  sql.SetAttribute("3", "BIT");
  sql.SetAttribute("4", "CPP ||  2+2");
  sql.ExecuteSql( );          // 输出是：ERROR SQL:  4,  CPP || 2+2,  is ungrammatical.
  return 0;
}
```

在 main 函数中，出现“？”的地方，都默认有一个整数值作为编号依次对应。成员函数 SetAttribute 通过编号一一对应赋值，并最终构造完整的 SQL 语句。

请按照上述 main 函数中对象 sql 调用各成员函数的方式来定义 CSqlStatement 类。成员函数 ExecuteSql 本意应是执行 sql 语句。但作为上机实验，改为输出所构造的整个 SQL 语句即可。

注意：1. 字符串类型只能使用自己定义的 CMyString 类，可以根据需要，自行增加类 CMyString 相关字符串操作的成员函数。注意：不能使用 C++系统提供的任何字符串类及库函数。

2. 请将 CMyString 类直接放在程序文件中，不用单独提交类 CMyString 的文件。

2. Define the class CList:

2.1 CList has member function, Insert(element, index), to insert an element into CList with index. If index is less than zero, the element will be the first.

2.2 You may define corresponding members of CList。

Here is a class, CStudent:

2.3 CStudent has two member variables: name, age;

2.4 Define other appropriate member functions if you need.

So they can be used as follows in main( ):

```
void main()
{
CStudent s1("Joan", 22), s2("John", 19), s3("Jean", 19);
CList<Student, 50> listStudent;                    // 50 is capcity of List
listStudent.Add(s1);
listStudent.Add(s2);
if (listStudent[0] == listStudent[1])              // Compare their ages
      cout << "They are the same age." << endl;
else
      cout << "They are not the same age." << endl;

listStudent.Instert(s2, 1);
if (listStudent[1] == listStudent[2])              // Compare their ages
      cout << "They are the same age." << endl;
else
      cout << "They are not the same age." << endl;
}
```

3. Here is the definition of class CShape：

```
class CShape
{
   public:
      virtual double Area() = 0;              //    area
      virtual double Volume() = 0;    //    volumn
};
```

As a base class, CShape, define its derived class , CCircle and CTriangle， to override the pure virtual functions in it.

So client can unify operation CCircle and CTtrangle through CAD class.

```
class CAD
{
   private:
      CShape& shape;
   public:
      CAD(Shape& S) : shape(S);
      double Area()      { return shape.Area(); }
      double Volume() { return shape.Volume(); }
};
```