1. 学校里的学生、教师基本信息包括姓名，性别，出生年月，ID（学生以入学年份开头，教师以 t 开头）。根据下面给出的示例性数据，定义相关的类及成员，并完成以下功能：

1) GetTeacherByCourse( )：学生能查询课程的授课教师;

2) GetScoreByCourse( )：学生根据课程名称查询成绩；

3) GetStudInfoByID( )：教师通过 ID 能知道学生的基本信息（包括：姓名、性别和年龄）；

4) SetScore2Course( )：教师给出课程的成绩；

要求：

1）上述功能的接口定义、所属类自行定义；

2）下面的示例数据，可以直接写在程序中；

3）本题需定义两个及以上的类完成相关功能，同时简要描述：所定义类之间的关系（组合、继承、依赖、实现等）及设计思路。

示例性的数据：

学生姓名	性别	出生年月	SID
李逵	男	2000.8	20200801
花千羽	女	2003.10	20221011
TF-BOY	男	2004.9	20210901
冷冰冰	女	2004.8	20221217

教师姓名	性别	出生年月	TID
孙悟空	男	1965.8	t20011211
张三丰	男	1979.10	t20060708

课程编号	课程名称	TID
COM002	自然语言理解	t19971211
COM006	面向对象技术与方法	t20060708
COM016	大数据处理	t20160708

学生 ID	课程编号	成绩
20200801	COM002	86
20221011	COM006	97
20210901	COM016	90
20221217	COM006	79

2. Define the class template CList:

2.1 CList has member functions such as Add and Remove;

2.2 You may define corresponding members of CList;

2.3 You CAN'T use STL(Standard Template Library) in C++.

Here is a class, CStudent:

2.3 CStudent has two member variables: name, age;

2.4 Define other member functions if you need.

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

```
int main()
{
    CStudent s1("Joan", 22), s2("John", 19)，s2("Joe", 22);
    CList<Student>  listStudent;
    listStudent.Add(s1);
    listStudent.Add(s2);
    listStudent.Add(s3);
    listStudent.Remove(1);                  // 1 is the 2rd element index of listStudent
    if (listStudent[0] == listStudent[1])   // If two students have same age.
        cout << "Equal." << endl;
    else
        cout << "Not equal." << endl;
    return 0;
}
```

Output is : Equal.

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

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

    return 0;
}
```

这种方法容易造成 SQL 语句的注入错误：如果 useName="user"，password = "abc || 1 == 1"，那么 where 字句中的密码查询结果都是 true。

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 语句的构成一般采用以下方法来构造：

```
int main( )
{
    CSqlStatement sql ="select ?, ? from student where SID = ?";   // 假定：sql 语句没有错误
    sql.SetAttribute("1", "Name");
    sql.SetAttribute("2", "Age");
    sql.SetAttribute("3", "2020007");    //  如果：sql.SetAttribute("3", "abc ||  2023 == 2023");
这时，成员函数应抛出异常：Errors in setting attribution
    sql.ExecuteSql( );

    return 0;
}
```

在 main 函数中，出现“？”的地方，都默认有一个整数编号依次相对应。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 调用各成员函数的形式，来定义 CSqlStatement 类。在实际应用中，成员函数 ExecuteSql 的功能是执行 sql 语句。但作为课堂实验，改为输出所构造的整个 SQL 语句。

注意：

1. 字符串类型只能使用自己定义的 CMyString 类，不能使用 C/C++ 提供的字符串类及库函数。
2. 请将 CMyString 类直接放在程序文件中，不要单独提交 CMyString 类的文件。

[operation 2]　用迭代器实现 for 语句的应用。有如下模板类的成员函数的声明：

```
template <class T>
class CMyVector
{
public:
	CMyVector();
	~CMyVector();
	void push_back(const T&);		// 新增元素，并放在 vecotr 的最后。
private:
	int next;	// 保存下一个元素的位置
	T* storage; // 动态分配的数组，"堆"的整个存储空间
};
```

要求：

1）实现该模板类的成员函数；

2）定义该模板类的迭代器 Iterator，并实现其构造函数、重载++、-- 为前缀及后缀运算符；

3）为了使得 for 语句能成功运行，至少在模板类中定义与迭代器有关的成员函数：begin()，end()，及重载运算符 !=；

4）所定义的所有类使得 main 函数能正确运行。

```
int main()
{
	// Handle int type
	CMyVector<int> intVector;
	for (int i = 0; i < 5; i++) { intVector.push_back(i); }

	// 1. 用迭代器方式，逆向输出数据
	for (CMyVector<int>::iterator it = intVector.end();  it != intVector.begin();  it--)
	{
		cout << *it << endl;
	}
	cout << endl;

	// 2. 用 for 语句方式，正向输出数据
	/*	为了满足 C++新的 for 语句标准，模板类 CMyVector 中必须定义迭代器，
		以及 begin、end 函数，和重载运算符 !=。	*/
	for (int value : intVector)
		cout << value << endl;

			return 0;
}
```