```
shared_ptr<int> makes(int value)
{
    //return new int(value);不可以，无法把new得到的int* 转换成shared_ptr
    return shared_ptr<int>(new int(value)); //可以，显式的用int *创建shared_ptr<int>
}
```

```
//三：shared_ptr基础
//共享所有权，不是被一个shared_ptr拥有，而是被多个shared_ptr之间相互协作；shared_ptr有额外开销；
//工作原理：引用计数，每个shared_ptr的拷贝都指向相同的内存。
    //所以，只有最后一个指向该内存（对象）的shared_ptr指针不需要再指向该对象时，那么这个shared_ptr才会去析构所
//最后一个指向该内存对象的shared_ptr在什么情况下会释放该对象（shared_ptr所指向的对象）呢？
//a)这个shared_ptr被析构的时候；
//b)这个shared_ptr指向其他的对象时；
//垃圾回收机制；我们从此不用担心对象何时被delete；
//类模板,用到<>，<>里，就是指针可以指向的类型 ，后边再跟智能指针名；
//格式：shared_ptr<指向的类型> 智能指针名；
```

```
//（3.1）常规初始化（shared_ptr和new配合）
//shared_ptr<int> pi; //指向int的智能指针，名字pi,但目前指向为空，空指针，nullptr;
//shared_ptr<int> pi(new int(100)); //pi指向一个值为100的int型数据
//shared_ptr<int> pi2 = new int(200); //不行，智能指针是explicit，不可以进行隐式类型转换。必须用直接初始化形式；
//shared_ptr<int> pi3 = makes(130);
//裸指针可以初始化shared_ptr，但不推荐。智能指针和裸指针不要穿插用；
/*int *pi = new int;
shared_ptr<int> p1(pi);

//shared_ptr<int> p1(new int);
```

```cpp
//三：性能说明
//shared_ptr的尺寸是裸指针的2倍；   weak_ptr尺寸裸指针的2倍；
char *p;
int ilenp = sizeof(p); //4字节
shared_ptr<string> p1;
int ilensp = sizeof(p1); //8字节，包含两个裸指针的

//a)第一个裸指针指向的是这个智能指针所指向的对象
//b)第二个裸指针 指向一个很大的数据结构（控制块），这个控制块里边有啥：
    //b.1)所指对象的强引用计数：shared_ptr
    //b.2)所指对象的弱引用计数：weak_ptr
    //b.3)其他数据，比如删除器指针，内存分配器；
//这个控制块是由第一个指向某个对象的shared_ptr来创建的；

//控制块创建时机：
//a)make_shared：分配并初始化一个对象，返回指向此对象的shared_ptr，所以，这个make_shared它总是能够创建一个控制块
```

```cpp
//四：补充说明和使用建议
//掌握了绝大部分shared_ptr用法；小部分没讲解，靠大家摸索。
//分配器，解决内存分配问题；
//shared_ptr<int> p((new int),mydeleter(),mymallocator<int>());..
```

```cpp
//（3.2）移动语义
shared_ptr<int> p1(new int(100));
shared_ptr<int> p2(std::move(p1)); //移动语义，移动构造一个新的智能指针对象p2，
                                   //p1就不再指向该对象（变成空），引用计数依旧是1；

shared_ptr<int> p3;
p3 = std::move(p2); //移动赋值， p2指向空，p3指向该对象，整个对象引用计数仍旧为1；

//移动肯定比复制块；复制你要增加引用计数，移动不需要；
//移动构造函数快过复制构造函数，移动赋值运算符快过拷贝赋值运算符；
```

```cpp
//四：补充说明和使用建议
//a)掌握了绝大部分shared_ptr用法；小部分没讲解，靠大家摸索。
//分配器，解决内存分配问题；
//shared_ptr<int> p((new int),mydeleter(),mymallocator<int>());..

//b)谨慎使用，凡是老师没讲到过的用法；
//new shared_ptr<T> ，memcpy()   奇怪用法，大家不要轻易尝试。

//c)优先使用make_shared()，不能让自己定义自己的删除器
shared_ptr<string> ps1(new string("I Love China!")); //分配两次内存
auto ps2 = make_shared<string>("I Love China!"); //分配1次内存
```