1. Write your own guard to manage handle returned by Win32 API function CreateFile (or open(…), if you use Linux)

2. Implement custom deleter for std::unique\_ptr in three different ways to close a SOCKET.

3. Implement simplified versions std::shared\_ptr and std::weak\_ptr. Use next interface in your implementation:

template <typename Resource>

class SharedPtr

{

SharedPtr();

explicit SharedPtr(Resource\* res);

SharedPtr(const SharedPtr<Resource>& rhs);

SharedPtr(const WeakPtr<Resource>& rhs);

SharedPtr<Resource>& operator=(const SharedPtr<Resource>& rhs);

~SharedPtr();

void Reset();

void Reset(Resource\* res);

void Swap(SharedPtr<Resource>& rhs);

Resource\* Get() const;

Resource& operator\*() const;

Resource\* operator->() const;

long UseCount() const;

};

template<typename Resource>

class WeakPtr

{

public:

WeakPtr();

WeakPtr(const WeakPtr<Resource>& rhs);

WeakPtr(const SharedPtr<Resource>& rhs);

WeakPtr<Resource>& operator=(const WeakPtr<Resource>& rhs);

WeakPtr<Resource>& operator=(const SharedPtr<Resource>& rhs);

~WeakPtr();

void Reset();

void Swap(WeakPtr<Resource>& rhs);

long UseCount() const;

bool Expired() const;

SharedPtr<Resource> Lock() const;

};

Please, keep classes declarations and methods definitions in a single file. Make sure, that your implementation can be used by just including that single file, i.e., next code should compile and work:

#include “YourImplementation.h”

int main()

{

SharedPtr<int> shared(new int(24));

WeakPtr<int> weak = shared;

}