به نام خدا

گزارش تمرین سری چهارم AP

محمدمهدی مالوردی

9723079

استاد جهانشاهی

توابع پیاده سازی شده در UniquePtr به صورت زیر می باشد:

include > C unique_ptr.h > ...

```h
#ifndef UNIQUE_PTR
#define UNIQUE_PTR

template<typename T>
class UniquePtr
{
    private :  T* _p;

public:
 UniquePtr();
 UniquePtr(T*);
 UniquePtr(const UniquePtr&) = delete;
 T* get();
 T& operator=(UniquePtr&) = delete;
 T& operator*();
 T* operator->();
 void reset();
 void reset(T*);
 T* release();
 ~UniquePtr();
 explicit operator bool();
};
template<typename T>
UniquePtr<T> make_unique(T ptr)
{
    return UniquePtr<T>{new T{ptr}};
}

#include "unique_ptr.hpp"
#endif //UNIQUE_PTR

```

فراخوانی توابع در UniquePtr.h

include > unique_ptr.hpp > UniquePtr()

```cpp
template <typename T>
UniquePtr<T>::UniquePtr()
{
    _p = nullptr;
}

template <typename T>
UniquePtr<T>::UniquePtr(T* p)
{
    _p = p;
}

template <typename T>
T* UniquePtr<T>::get(){
    return _p;

}
template <typename T>
T& UniquePtr<T>::operator*()
{
    return (*_p);
}
template <typename T>
T* UniquePtr<T>::operator->()
{
    return _p;
}
template <typename T>
void UniquePtr<T>::reset(){
    delete _p;
    _p = NULL;
}
template <typename T>
void UniquePtr<T>::reset(T* ptr){
    delete _p;
    _p = ptr;

}
template <typename T>
T* UniquePtr<T>::release()
{
    T* rawPtr = _p;
    _p = NULL;
    return rawPtr;
}
template <typename T>
UniquePtr<T>::~UniquePtr()
{
    if(_p)
        delete _p;
    _p = NULL;
}

```

```cpp
}

template <typename T>
UniquePtr<T>::operator bool(){
    if(_p == nullptr) return false;
    else
    return true;
}

```

تعریف توابع در UniquePtr.hpp

```cpp
#ifndef SHARED_PTR
#define SHARED_PTR

template <typename T>
class SharedPtr
{

private:
T* _p = nullptr;
int* refCount = nullptr;
public:
 SharedPtr();
 SharedPtr(T*);
 SharedPtr(const SharedPtr&);
 SharedPtr& operator=(const SharedPtr<T>&);
 T* get();
 int use_count() const;
 T& operator*();
 T* operator->();
 void reset();
 void reset(T*);
 ~SharedPtr();
 explicit operator bool();
};
 template <typename T>
SharedPtr<T> make_shared(T ptr)
{
    return SharedPtr<T>{new T{ptr}};
}

#include "shared_ptr.hpp"
#endif //SHARED_PTR
```

فراخوانی توابع در SharedPtr.h

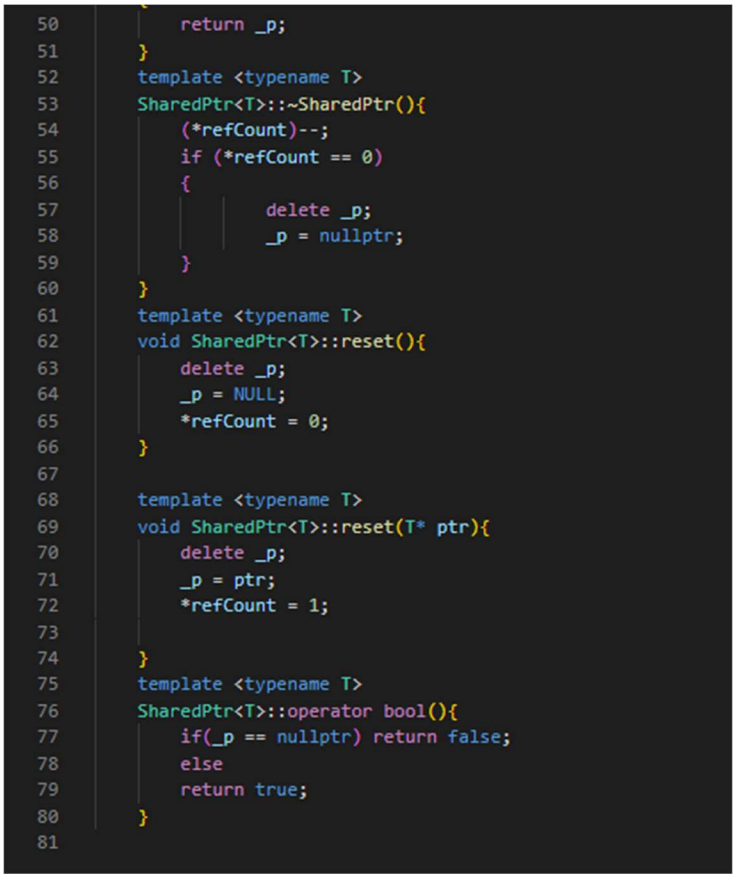
include > shared_ptr.hpp > operator=(const SharedPtr<T>&)

```cpp
template <typename T>
    SharedPtr<T>::SharedPtr() :_p(nullptr), refCount(new int(0)){}


    template <typename T>
    SharedPtr<T>::SharedPtr(T* ptr) : _p(ptr), refCount(new int(1)){}


    template <typename T>
    SharedPtr<T>::SharedPtr(const SharedPtr& obj)
    {
        this->_p = obj._p;
        this->refCount = obj.refCount;
        if (nullptr != obj._p)
        {
            (*this->refCount)++;
        }
    }
    template <typename T>
    SharedPtr<T>& SharedPtr<T>::operator=(const SharedPtr<T> & obj)
    {
        if (_p == obj._p)
        return *this;
        _p = obj._p;
        refCount = obj.refCount;
        if (nullptr != obj._p)
        {
            (*refCount)++;
        }
        return *this;
    }
    template <typename T>
    T* SharedPtr<T>::get(){
        return _p;

    }
    template <typename T>
    int SharedPtr<T>::use_count() const
    {
        return *refCount;
    }
    template <typename T>
    T& SharedPtr<T>::operator*()
    {
        return (*_p);
    }
    template <typename T>
    T* SharedPtr<T>::operator->()
    {
        return _p;
    }
```



```
50          return _p;
51      }
52      template <typename T>
53      SharedPtr<T>::~SharedPtr(){
54          (*refCount)--;
55          if (*refCount == 0)
56          {
57                  delete _p;
58                  _p = nullptr;
59          }
60      }
61      template <typename T>
62      void SharedPtr<T>::reset(){
63          delete _p;
64          _p = NULL;
65          *refCount = 0;
66      }
67
68      template <typename T>
69      void SharedPtr<T>::reset(T* ptr){
70          delete _p;
71          _p = ptr;
72          *refCount = 1;
73
74      }
75      template <typename T>
76      SharedPtr<T>::operator bool(){
77          if(_p == nullptr) return false;
78          else
79          return true;
80      }
81
```

تعریف توابع در SharedPtr.hpp

توابع همه مطابق صورت سوال نوشته شد. تابع نوشته شده برای چالش هر دو بخش نیز operator bool می باشد که درون آن مطابق خواسته صورت سوال نوشته شد.

باید دقت شود که در فایل های hpp. قبل از هر تابع template <typename T> را نیز بنویسیم.

همچنین include هر فایل hpp. باید در انتهای فایل های h. نوشته شود.

در نهایت هر 21 تست پاس شدند.

لینک گیت هاب:

https://github.com/mahdimld/HW04