

به نام خدا

گزارش تمرین سری چهارم AP

محمد مهدی مالموردی

9723079

استاد جهانشاهی

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

```
include > C unique_ptr.h > ...
1  #ifndef UNIQUE_PTR
2  #define UNIQUE_PTR
3
4  template<typename T>
5  class UniquePtr
6  {
7      private :   T* _p;
8
9      public:
10     UniquePtr();
11     UniquePtr(T*);
12     UniquePtr(const UniquePtr&) = delete;
13     T* get();
14     T& operator=(UniquePtr&) = delete;
15     T& operator*();
16     T* operator->();
17     void reset();
18     void reset(T*);
19     T* release();
20     ~UniquePtr();
21     explicit operator bool();
22 };
23 template<typename T>
24 UniquePtr<T> make_unique(T ptr)
25 {
26     return UniquePtr<T>{new T{ptr}};
27 }
28
29 #include "unique_ptr.hpp"
30 #endif //UNIQUE_PTR
31
```

فراخوانی توابع در UniquePtr.h

```

include > unique_ptr.hpp > UniquePtr()

1  template <typename T>
2  UniquePtr<T>::UniquePtr()
3  {
4      _p = nullptr;
5  }
6
7  template <typename T>
8  UniquePtr<T>::UniquePtr(T* p)
9  {
10     _p = p;
11 }
12
13 template <typename T>
14 T* UniquePtr<T>::get(){
15     return _p;
16 }
17
18 template <typename T>
19 T& UniquePtr<T>::operator*()
20 {
21     return (*_p);
22 }
23
24 template <typename T>
25 T* UniquePtr<T>::operator->()
26 {
27     return _p;
28 }
29
30 template <typename T>
31 void UniquePtr<T>::reset(){
32     delete _p;
33     _p = NULL;
34 }
35
36 template <typename T>
37 void UniquePtr<T>::reset(T* ptr){
38     delete _p;
39     _p = ptr;
40 }
41
42 template <typename T>
43 T* UniquePtr<T>::release()
44 {
45     T* rawPtr = _p;
46     _p = NULL;
47     return rawPtr;
48 }
49
50 template <typename T>
51 UniquePtr<T>::~~UniquePtr()
52 {
53     if(_p)
54         delete _p;
55     _p = NULL;
56 }

```

```

52     }
53
54     template <typename T>
55     UniquePtr<T>::operator bool(){
56         if(_p == nullptr) return false;
57         else
58             return true;
59     }
60

```

تعریف توابع در UniquePtr.hpp

```

C shared_ptr.h 1 X
include > C shared_ptr.h > ...
1  #ifndef SHARED_PTR
2  #define SHARED_PTR
3
4  template <typename T>
5  class SharedPtr
6  {
7
8  private:
9      T* _p = nullptr;
10     int* refCount = nullptr;
11     public:
12         SharedPtr();
13         SharedPtr(T*);
14         SharedPtr(const SharedPtr&);
15         SharedPtr& operator=(const SharedPtr<T>&);
16         T* get();
17         int use_count() const;
18         T& operator*();
19         T* operator->();
20         void reset();
21         void reset(T*);
22         ~SharedPtr();
23         explicit operator bool();
24     };
25     template <typename T>
26     SharedPtr<T> make_shared(T ptr)
27     {
28         return SharedPtr<T>{new T{ptr}};
29     }
30
31     #include "shared_ptr.hpp"
32     #endif //SHARED_PTR

```

فراخوانی توابع در SharedPtr.h

```

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

1  template <typename T>
2      SharedPtr<T>::SharedPtr() : _p(nullptr), refCount(new int(0)){}
3
4
5      template <typename T>
6      SharedPtr<T>::SharedPtr(T* ptr) : _p(ptr), refCount(new int(1)){}
7
8
9      template <typename T>
10     SharedPtr<T>::SharedPtr(const SharedPtr& obj)
11     {
12         this->_p = obj._p;
13         this->refCount = obj.refCount;
14         if (nullptr != obj._p)
15         {
16             (*this->refCount)++;
17         }
18     }
19     template <typename T>
20     SharedPtr<T>& SharedPtr<T>::operator=(const SharedPtr<T> & obj)
21     {
22         if (_p == obj._p)
23             return *this;
24         _p = obj._p;
25         refCount = obj.refCount;
26         if (nullptr != obj._p)
27         {
28             (*refCount)++;
29         }
30         return *this;
31     }
32     template <typename T>
33     T* SharedPtr<T>::get(){
34         return _p;
35     }
36
37     template <typename T>
38     int SharedPtr<T>::use_count() const
39     {
40         return *refCount;
41     }
42     template <typename T>
43     T& SharedPtr<T>::operator*()
44     {
45         return (*_p);
46     }
47     template <typename T>
48     T* SharedPtr<T>::operator->()
49     {
50         return _p;
51     }

```

```

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>