## Example Program for C++ Smart Pointers

```
#include<memory>
#include<iostream>

struct A;
struct B;

struct A
{
    std::shared_ptr<B> b;
    ~A() { std::cout << "~A()\n"; }
};

struct B
{
   std::shared_ptr<A> a;
    ~B() { std::cout << "~B()\n"; }
};

struct C
{
     std::shared_ptr<A> a;
    ~C() { std::cout << "~C()\n"; }
};

struct D;
struct E;

struct D
{
     std::shared_ptr<E> e;
    ~D() { std::cout << "~D()\n"; }
};

struct E
{
     std::weak_ptr<D> d;
    ~E() { std::cout << "~E()\n"; }
};

struct F
{
    std::shared_ptr<E> e;
    ~F() { std::cout << "~F()\n"; }
};



int main()
{
```

```
//Example std::unique_ptr

std::unique_ptr<int> p1 (new int);
*p1=10;
std::cout<< *p1<<std::endl;  //10
std::unique_ptr<int> p2;
p2 =std::move(p1); // p1 is removed p2 only exists =>move constructor
std::cout<< *p2<<std::endl; //10
if(static_cast<bool>(p1))
std::cout<<"p1 Valid"<<std::endl;
else
std::cout<<"p1 Not Valid"<<std::endl;  // p1 Not Valid
if(static_cast<bool>(p2))
std::cout<<"p2 Valid"<<std::endl; // p2 Valid
else
std::cout<<"p2 Not Valid"<<std::endl;

int i;
std::unique_ptr<int[]> arrptr(new int[5]);
for(i=0;i<5;i++)
{
arrptr[i]=i;
}

for(i=0;i<5;i++)
{
std::cout<<arrptr[i]<<std::endl;  // 0 1 2 3 4
}

//Example std::shared_ptr

std::shared_ptr<int>p3(new int);
*p3=20;
std::shared_ptr<int>p4(p3); //copy constructor
std::cout<< *p3<<std::endl;  // 20
std::cout<< *p4<<std::endl; // 20
std::cout<< p3.use_count()<<std::endl; //2
std::cout<< p4.use_count()<<std::endl; //2
std::shared_ptr<int>p5(std::move(p3)); //move constructor

std::cout<< *p5<<std::endl; // 20
std::cout<< p3.use_count()<<std::endl; // 0
std::cout<< p4.use_count()<<std::endl; // 2
std::cout<< p5.use_count()<<std::endl; // 2
std::shared_ptr<int>p6(new int);
*p6=30;
std::shared_ptr<int>p7;
p7=p6;//copy assignment
std::cout<< *p6<<std::endl; // 30
std::cout<< *p7<<std::endl; // 30
```

```
p6=std::make_shared<int>(40); //move assignment can be directly used without creating p7
std::cout<< *p6<<std::endl; // 40
std::cout<< *p7<<std::endl; // 30
std::cout<< p6.use_count()<<std::endl; // 1
std::cout<< p7.use_count()<<std::endl; // 1

//Example std::weak_ptr

std::shared_ptr<int> p8,p9;
std::weak_ptr<int> weakp;
p8=std::make_shared<int>(50);
std::cout<< *p8<<std::endl; // 50
std::cout<<"praveen"<<std::endl;
weakp=p8;
p9=weakp.lock();
std::cout<< *p9<<std::endl; // 50
p9.reset();
std::cout<< *p8<<std::endl; // 50
p9=weakp.lock();
std::cout<< *p8<<std::endl; // 50
std::cout<< *p9<<std::endl; // 50
p8.reset();
p9.reset();
p8=weakp.lock();
if(static_cast<bool>(p8))
std::cout<<"p8 Valid"<<std::endl;
else
std::cout<<"p8 Not Valid"<<std::endl; //Becasue no shared pointer owns int  // p8 Not Valid
if(static_cast<bool>(p9))
std::cout<<"p9 Valid"<<std::endl;
else
std::cout<<"p9 Not Valid"<<std::endl; //Becasue no shared pointer owns int // p9 Not Valid

//Example std::shared_ptr will cause problem during circular referencing

    auto a = std::make_shared<A>();
    auto b = std::make_shared<B>();
    auto c = std::make_shared<C>();

    // Circular reference
    a->b = b;
    b->a = a;

    // Third resource
    c->a = a;

//Example std::weak_ptr will not cause problem during circular referencing

    auto d = std::make_shared<D>();
    auto e = std::make_shared<E>();
    auto f = std::make_shared<F>();
```

```
        // Circular reference
        d->e = e;
        e->d = d;
        // Third resource
        f->e = e;
return 0;
}
```

**Output**

```
10

10

p1 Not Valid

p2 Valid

0

1

2

3

4

20

20

2

2

20

0

2

2

30

30

40

30

1

1

50
```

```
praveen
50
50
50
50
p8 Not Valid
p9 Not Valid
~F()
~D()
~E()
~C()
```