```
// Adapted from Reference: Bjarne Stroustrup. 2014. Programming: Principles and Practice Using C++
(2nd. ed.).

// Addison-Wesley Professional, pg. 644-645.

#include <iostream>

#include <memory>

#include <vector>

using namespace std;


struct X {

  int val;

  void out(const string& s, int nv) {

     cerr << this << "->" << s << ":" << val << "(" << nv << ")\n";

  }


  X(){out("X()",0); val = 0;}  // default constructor

  X(int v) :val(v) {out("X(int)",v);}


  X(const X& x) :val(x.val) {out("X(X&)",x.val);}   // copy constructor


  X& operator=(const X&a) {    // copy assignment operator

    out("X::operator=()", a.val); val=a.val; return *this;

  }


  ~X() {out("~X()",0);}  // destructor

  };

0) X glob(2); // Global variable

  X copy(X a) { return a;}

  X copy2(X a) { X aa = a;  return aa;}
```

```
X& ref_to(X& a) {return a;}

unique_ptr<X> make(int i) {X a(i); return make_unique<X>(a);}

struct XX {X a; X b;};

// Trace what is output by main.
// What is printed to std error?  You can run it and see.
// What function is called by each statement?
int main() {
1)  X loc{4};  // local variable
2)  X loc2{loc}; // copy construction
3)  loc = X{5}; // copy assignment
4)  loc2 = copy(loc); // call by value and return;
5)  loc2 = copy2(loc);
6)  X loc3{6};
    X& r = ref_to(loc);
7)  unique_ptr<X> p1 = make_unique<X>(7);
8)  p1.reset(); // delete the X from the heap
9)  p1 = make_unique<X>(8);
10) p1.reset(); // delete the X from the heap
11) vector<X> v(4);
12) XX loc4;
13) p1 = make_unique<X>(9);  // create X on heap and then delete it
14) p1.reset();
15) unique_ptr<X[]> p2 = make_unique<X[]>(5); //create array of X on heap and delete
16) p2.reset();
}
```

Trace output

0) 0x604214->X(int):2(2) glob constructor

1) 0x7fff773a9fa0->X(int):4(4) loc constructor

2) 0x7fff773a9fb0->X(X&):4(4) loc2 copy constructor

3) 0x7fff773a9fc0->X(int):5(5) constructor for temp X

3) 0x7fff773a9fa0->X::operator=():4(5) copy assignment into loc

3) 0x7fff773a9fc0->~X():5(0) destructor for temp X

4) 0x7fff773a9fd0->X(X&):5(5) copy constructor for input param

4) 0x7fff773a9fe0->X(X&):5(5) copy constructor for return value

4) 0x7fff773a9fb0->X::operator=():4(5) copy assignment for return value to loc2

4) 0x7fff773a9fe0->~X():5(0) destructor for return value

4) 0x7fff773a9fd0->~X():5(0) destructor for input param

5) 0x7fff773a9ff0->X(X&):5(5) copy constructor for input param

5) 0x7fff773aa000->X(X&):5(5) copy constructor for return value

5) 0x7fff773a9fb0->X::operator=():5(5) copy assignment to loc2

5) 0x7fff773aa000->~X():5(0) destructor for return value

5) 0x7fff773a9ff0->~X():5(0) destructor for return value

6) 0x7fff773aa010->X(int):6(6) constructor for loc3

* → notice no calls or prints for ref-to *

7) 0x1a3fc20->X(int):7(7) → constructor for X on heap

8) 0x1a3fc20->~X():7(0) → destructor for X on heap

9) 0x1a3fc20->X(int):8(8) → constructor for X on heap

10) 0x1a3fc20->~X():8(0) → destructor for X on heap

11) 0x1a3fc20->X():0(0)

11) 0x1a3fc24->X():0(0)

11) 0x1a3fc28->X():0(0)

11) 0x1a3fc2c->X():0(0)

constructor for vector will call constructor for each of 4 X objects

12) 0x7fff773aa040->X():2000331088(0)

12) 0x7fff773aa044->X():32767(0)

constructor for members of XX loc4

13) 0x1a3fc40->X(int):9(9) – constructor for X on heap

```
14) 0x1a3fc40->~X():9(0)
15) 0x1a3fc68->X():0(0)
15) 0x1a3fc6c->X():0(0)
15) 0x1a3fc70->X():0(0)
15) 0x1a3fc74->X():0(0)
15) 0x1a3fc78->X():0(0)
16) 0x1a3fc78->~X():0(0)
16) 0x1a3fc74->~X():0(0)
16) 0x1a3fc70->~X():0(0)
16) 0x1a3fc6c->~X():0(0)
16) 0x1a3fc68->~X():0(0)
    0x7fff773aa044->~X():0(0)
    0x7fff773aa040->~X():0(0)
    0x1a3fc20->~X():0(0)
    0x1a3fc24->~X():0(0)
    0x1a3fc28->~X():0(0)
    0x1a3fc2c->~X():0(0)
    0x7fff773aa010->~X():6(0)
    0x7fff773a9fb0->~X():5(0)
    0x7fff773a9fa0->~X():5(0)
    0x604214->~X():2(0)
```

destructor for X on heap

To create array of X on heap constructor called for each element X

destructor is called for each X element of array on heap

— objects on stack destroyed in reverse order

} destructor loc4

destructor for vector must destroy each X element

→ destructor for loc3

→ destructor for loc2

→ destructor for loc

→ destructor for glob