```
// 368 Worksheet: Unique Pointers
#include <iostream>
#include <              >
using namespace std;
struct Coord{
  float x; float y;
  ~Coord() {cout << "Destroy Coord: " << x << " " << y << "\n";}
};
class Triangle {
  Coord A;
  Coord* B = nullptr;
  unique_ptr<Coord> C = nullptr;
public:
  Triangle(float x1, float y1, float x2, float y2,
           float x3, float y3)
    : A(x1, y1), B(new Coord(x2, y2)), C(new Coord(x3, y3)) {}
  ~Triangle() {

  }
};
int main() {
  auto t1 = Triangle(1,1,2,2,3,4);
  auto t2 = new Triangle(0,0,1,5,2,0);
  auto t3 = make_unique<Triangle>(7,7,8,9,6,6);

}
```

1. Complete the above include so we can use unique_ptr
2. Which coordinates will be released (✓) vs. leaked (✘)?

   {1,1} {2,2} {3,4}   {0,0} {1,5} {2,0}   {7,7} {8,9} {6,6}

3. Add code above so that we release/destroy all nine Coord objects.
4. Cross out lines the compiler won't allow? Consider each individually.

```
Triangle* other = &t1;
Triangle* other = t2;
Triangle* other = t3;
Triangle* other = &t3;
unique_ptr<Triangle> other = t3;
Triangle* other = t3.get();
```

```
// 368 Worksheet: Shared Pointers
```

```
1   struct Coord{
2     float x; float y;
3     ~Coord() {
4       cout<<"Bye "<<x<<" "<<y<<"\n";
5     }
6   };
7
8   int main() {
9     Coord A{1,1};
10    Coord* B;
11    {
12      Coord* C = new Coord(7,8);
13      cout << C->x << "\n";
14      B = C;
15
16    }
17    cout << B->x << "\n";
18
19    auto D = make_shared<Coord>(3,3);
20    cout << D.use_count() << "\n";
21
22    cout << B->y << "\n";
23
24
25    {
26      auto E = D;
27      cout << D.use_count() << "\n";
28
29    }
30    cout << D.use_count() << "\n";
31
32
33  }
```

Output:

5. Add any delete calls necessary, at the soonest line(s) possible.
6. Write any output on the right hand side (based on the modified code).