**Practicing Pointers**

1. Consider the following code segment. Make sure to include <memory>

auto a = make\_shared<int>(7);

auto b = a;

* 1. What would display if you did a cout statement for a? How about b? **A would display the address to the integer with value 7. B would also display that as it is another pointer pointing to the same thing with shared\_ptr.**
  2. How would you display the value of the b pointer? **Cout << \*b**

1. Consider the following two code segments

| **Shared Pointer** | **Unique Pointer** |
| --- | --- |
| auto one = make\_shared<int>(5);  auto copy1 = one; | auto two = make\_unique<int>(6);  auto copy2 = two; |

* 1. Does the shared pointer give an error message? **No, as both the variables can share the address**
  2. Does the unique pointer give an error message? **Yes, because a pointer is already pointing to that address.**
  3. What is happening that is causing the code to receive an error message? **The make\_unique function has a default\_delete function that deletes itself after initialization so that the next variable cannot reference the same address.**

1. Take a look at the following code segment.

| void copyPointer(unique\_ptr<int>b){  //Do something  }  int main()  {  auto a = make\_unique<int>(7);  copyPointer(a);  ... |
| --- |

* 1. Explain the error with the pointers? **The function “std::unique\_ptr” can’t be called because it is a deleted function, going back to how unique\_ptr delete themselves by default.**
  2. What type of pointer should be used instead? **A shared pointer.**

1. Consider the following code segment…

auto num = make\_unique<int>(10);

auto lst = make\_unique<int[]>(10);

for(int i = 0; i < 10; i++)

lst[i] = i;

Which of the following statements would compile, and what would be displayed? **All of them would compile except e because e is a dereferenced list, which doesn’t have a way to output to cout. num would return its address, as would lst, lst[5] would give the 6th value of the lst array, and \*num would give 5, the dereferenced value.**

* 1. cout << num << endl;
  2. cout << lst << endl;
  3. cout << lst[5] << endl;
  4. cout << \*num << endl;
  5. cout << \*lst << endl;

1. Practice pointers by writing the following statements
   1. Declare a shared pointer named *num* that references the int value 34.
      * auto num = make\_shared<int>(34);
   2. Declare a unique pointer array named *lst* size 3 of int values
      * auto lst = make\_unique<int[]>(3);
   3. Write the statement that sets position 0 to be the value 42
      * lst[0] = 42;
   4. Write the statement that sets position 1 to be the value in *num*
      * lst[1] = \*num;
   5. Write the statement that sets position 2 to be the value of position 1 added with the value in position 0
      * lst[2] = lst[1] + lst[0];