LAB GOALS

- Understanding Pointers
- Understanding Static vs. Dynamic Allocation
- Understanding the (Dot) vs. (Arrow) notation.

Step 1: Start with a simple C++ console project called I308_Lab1.

Step 2: Add a new (main.cpp) item (file) under the source files.
Step 3: Now create a new class called aNode. (make sure to include the necessary header files, and namespaces)

```cpp
class aNode {
public:
    char data;
    aNode *next;

    aNode(char mydata) { // Constructor
        data = mydata;
        next = NULL;
    }
};
```

Step 4: Now create a main() function and instantiate an object of type aNode.

```cpp
int main()
{
    aNode myNode_1('A'); // Instantiate an object of type aNode
    cout << myNode_1.data << endl;
    cout << myNode_1.next << endl;
    getchar(); // To pause the program
}
```

Step 5: Compile and run the program to make sure the above steps are working properly.

The above is an example of **statically allocating an object**, and then using the “dot” notation to access and display its properties.

Step 6: Now add the following code (above the getchar() line) and note the syntactic difference with step 5, and then note the results when you run the program.

```cpp
int main()
{
    // --------------------------
    aNode *myNode_2; // Create a pointer to aNode
    myNode_2 = new aNode('B'); // Instantiate an object of type aNode (call the constructor)
    cout << myNode_2->data << endl;
    cout << myNode_2->next << endl;
    getchar(); // To pause the program
}
```
Step 7: Compile and run the program to make sure the above steps are working properly.

The above is an example of **dynamically allocating an object**, and then using the “arrow” notation to access and display its properties.

Step 8: Now let’s try to understand pointers a little better. As you know, the above nodes are created in memory, and they have a physical address. So, let’s try to find out what is the actual memory location for these two nodes.

Please note the syntactic difference between trying to print the address of a statically allocated node vs. one that was dynamically allocated. We use the (&) or (address of) operator to get at the address of myNode_1. However Node2 was already a pointer (an address) so we can simply print it!

Step 9: Now let’s try to manually build a link list with the following 3 nodes. We need a head pointer which will eventually point to (myNode_1), and we also need third node (myNode_3) which we will create “statically” and then we will connect everything together.
Step 10: To make sure the linked list is correctly connected, we need to traverse it. So, let's add the following code and run the program again.

On your own:

```cpp
// Now let's learn more...Can you predict the output of the following lines?
cout << Head << endl;
cout << Head->next << endl;
cout << Head->next->next << endl;
cout << Head->next->next->next << endl;
cout << Head->data << endl;
cout << Head->next->data << endl;
cout << Head->next->next->data << endl;
```