Note: This exercise is to be done in **groups.** Submission is a .zip file.

Create a C++ program that implements `struct`-based containers for integers without using arrays.

1. (2 pts) Implement an `IntNode` struct, which should just contain a piece of data (i.e., your integer), along with pointers to other nodes as you deem necessary.

   For this exercise, `IntNode` should contain only variables (no functions).

2. (5 pts) Implement an `IntList` struct that has functions for:
   a. Creating the first node of a linked list (return a pointer to the created `IntNode`)
   b. Inserting a node after any other node (pass the pointer of the other `IntNode` to this function; return a pointer to the new `IntNode`)
   c. Getting the first node (return a pointer, or `NULL` if it does not exist)
   d. Getting the node after any other node (return `NULL` if it does not exist)
   e. Deleting a node

   Your main function should demonstrate creation and traversal of the linked list using these 5 functions. You may include additional functions as you see fit, but the above 5 functions are required.

3. (3 pts) Implement an `IntStack` struct that internally uses `IntNode` and has functions for push, pop, and current size. You should NOT expose the `IntNode` struct to the user. (i.e., the push and pop functions should work exclusively with primitive integers).

   Your main function should demonstrate the 3 stack functions that you implemented.
   Notes:
   - You do not actually need to (re)use `IntList` for this part – but if you do reuse `IntList`, do not expose the `IntList` to the user.
   - Your program should not crash or exhibit undefined behavior if you try to pop an empty stack. (It is up to you how to handle it.)

4. (2 pts) Your implementation should NOT cause memory leaks. (Refer to the individual exercise so you have an idea of how to check for memory leaks.)

Limitations for this exercise:

- Do NOT use the built-in list/iterator templates (or any other built-in template) of C++
- Do NOT use the inheritance features of C++
- (These are separate topics in their own right.)

Submit a zip file containing the following:

- <u>fully-commented</u> code: `Lab3_(surnames in alphabetical order)_code.cpp`
- header file (if any): `Lab3_(surnames in alphabetical order)_code.h`
- accomplished COA: `Lab3_(surnames in alphabetical order)_COA.pdf`
- name the zip file: `Lab3_(surnames in alphabetical order).zip`

Example submission (the other files are contained within the zip file):

- `Lab3_EspinosaManaogSanchez.zip`
    - `Lab3_EspinosaManaogSanchez_code.cpp`
    - `Lab3_EspinosaManaogSanchez_code.h`
    - `Lab3_EspinosaManaogSanchez_COA.pdf`

This lab will have a **defense component**. (8 pts) You will need to sign up for your defense slot, as discussed by your instructor.