**Name:** Ahmed Abdelghany  
**Lab Name:** Node Cycles and weak\_ptrs

**1. Circular Linked List Example (6 Nodes with weak\_ptr)**

I created a circular linked list using six nodes: **zero**, **one**, **two**, **three**, **four**, and **five**.  
Each node holds a weak\_ptr to the next node to **prevent memory cycles**.

The structure used:

struct Node {

string name;

weak\_ptr<Node> next;

Node(string name) : name{name} {}

~Node() { cout << "Node [" << name << "] destructor" << endl; }

};

The list is built in a circular fashion:  
**zero → one → two → three → four → five → zero**.

**2. Demonstrating Behavior with shared\_ptr vs weak\_ptr**

**Before Deletion:**

* **SERVER** traverses and prints nodes using shared\_ptrs.
* **CLIENT** traverses and prints nodes using weak\_ptrs.

Example output:

[zero] use\_count: 3 address: 0x...

[one] use\_count: 2 address: 0x...

[two] use\_count: 2 address: 0x...

...

**After Deletion:**

* **SERVER** clears all shared\_ptrs (releasing ownership).
* **CLIENT** tries to traverse using weak\_ptrs but finds expired pointers.
* **CLIENT** output after deletion:

Yipes! shared\_ptr not available

Yipes! shared\_ptr not available

...

* **Node destructors** are called safely, confirming proper memory deallocation without leaks.

**3. Observations**

* Building a circular linked list with only shared\_ptrs would cause a memory leak because the nodes reference each other, preventing automatic destruction.
* Using weak\_ptr correctly breaks the ownership cycle, allowing nodes to be destroyed safely.
* After the SERVER deletes the list, the weak\_ptrs correctly detect the expired nodes (via lock() returning null).
* This demonstrates **best practices in modern C++ memory management**, ensuring both **memory safety** and **program correctness** in cyclic data structures.

**4. GitHub Repository**

A GitHub-ready zipped folder is included containing([link](https://github.com/Ventapa/CircularLinkedList-WeakPtr.git)):

* Node.h — Declaration of the Node struct.
* LinkedList.h — Declaration of the LinkedList class.
* LinkedList.cpp — Implementation of linked list methods.
* main.cpp — Testing and demonstration program.
* LinkedList\_test.cpp — Basic unit tests for the project.
* README.md — Build and run instructions.