Data Structure Lab Assignment (CS 2172)
Assignment 5 Extension: Singly, Circular and Doubly Linked List

Time: Two weeks but not limited to ....

1. Develop an ADT using Linked List to represent a large number. Develop the following supports -

   a) Print the number in decimal (and hexadecimal possible) format
   b) Subroutine to add and deduct two numbers

2. Develop an ADT to represent a polynomial using a linked list. Also, develop the following subroutines –

   a) Display polynomial
   b) Add and deduct two polynomials

3. In a game, let us consider that N (user's input) number of persons are standing on a circle boundary and waiting to be evicted. Let's call the ID of each person 1, 2, 3, … N. Counting begins at a specified point (randomly selected) on the boundary and proceeds around the circle in a specified direction. A person is evicted after a specified number of people (S) is counted to be skipped. The procedure is repeated with the remaining people, starting from the person next to the evicted one, going in the same direction, and skipping the same number of people until only one person remains. The remaining person is selected as the winner of the game.

   Write a program for this game. This program aims to find the names of the persons evicted one after another and, finally, the winner's name. To develop the game, you must use a circular linked list where each node of the linked list represents a person.

4. Modify the code of Problem 3 (keep the copy of Problem 3 unchanged) with the following enhancements:
   a) Generates the value of S randomly in each iteration.
   b) If the value of S is positive, then the direction of proceeding around the circle is clockwise; otherwise, it is anticlockwise.

5. Design and implement a module to represent a memory pool of **linked nodes** with the following `listNode` declarations and APIs.

   Concept is required to implement this to be discussed based on your interest.

```
struct listNode {
    int data;
    struct listNode * next;
};
```

   `int createMemoryPoolNodes(...)` – A finite number of nodes will be allocated using `malloc(...)` system call and should be arranged as a circular linked list. This list should act as an available list.

   `int releaseMemoryPoolNodes(...)` – It releases all the nodes of the memory pool using `free(...)` system call. It should also log the memory leaks. For example, if the original pool was created with M nodes, and while releasing the pool, it has only N nodes, where M > N, then the number of node leaks is (M-N).

   `struct listNode *getNode()` – Allocates a node from the available list and returns a pointer to the node.

   `void releaseNode(struct listNode *)` – Accepts the pointer to the node and adds the node back to the available list.

   Now modify the Assignment-4 code (keep the copy of Assignment-4 unchanged) to use `getNode()` and `releaseNode()` instead of `malloc()` and `free()`. Also call `createMemoryPoolNodes()` and the beginning of main() and call `releaseMemoryPoolNode()` at the end of it. Finally, test the code to see code functioning properly. Check if there are any node leaks.