Institute for Advancing Intelligence, TCG CREST
(TCG Centres for Research and Education in Science and Technology)

Introduction to Programming and Data Structures
Ph.D. Coursework: First year, First Semester (Session: 2024-25)
Assignment #03

Full Marks: 200	Instructor: Dr. Laltu Sardar
Clarification Deadline: **2024-Sep-16**	Submission Deadline: **2024-Sep-18**

Instructions

- Use dynamic memory allocation where necessary, and ensure all dynamically allocated memory is freed appropriately.
- For input/output from/to a file, it is sufficient to use “r” and “w” mode.
- You can not use `<string.h>` library
- Please discuss, if necessary, with others but do not share your code or any part it.

Problems and Examples

Problem 1: Cycle on Linked Lists

1. **Create a linked list (Insertion at the end):**
   Write a C function to create a singly linked list by inserting nodes at the end of the list. Each node should contain a data element and a pointer to the next node.

   Example:
   Input sequence: $\{1, 2, 3, 4, 5\}$
   Output: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow$ NULL

2. **Create a cycle in the linked list:**
   Write a function that takes as input a value and the head of the linked list. If the value is found in the list, link the last node to the node containing that value to create a cycle. If the value does not exist in the list, no cycle should be created.

   Example:
   Input: Linked list: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow$ NULL, Value to link: 3
   Output: Linked list with cycle: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 3$ (cycle starts again at 3)

3. **Detect a cycle in the linked list:**
   Write a function to detect if there is a cycle in a given linked list.

   Example:
   Input: Linked list with cycle as created above
   Output: Cycle detected.

Problem 2: Remove Duplicates from a Sorted Linked List

1. **Create a linked list:**
   Write a function to create a singly linked list by inserting elements at the end.

   Example:
   Input sequence: {1, 3, 3, 5, 5, 7}
   Output: 1 → 3 → 3 → 5 → 5 → 7 → NULL

2. **Sort the linked list:**
   After creating the linked list, write a function to sort the elements in the list in ascending order. (This step assumes the list may not be sorted initially.)

   Example:
   Input sequence: {5, 3, 1, 7, 3}
   Output: 1 → 3 → 3 → 5 → 7 → NULL

3. **Remove duplicate elements:**
   Once the linked list is sorted, write a function to remove all duplicate elements, ensuring each element appears only once in the list.

   Example:
   Input: 1 → 3 → 3 → 5 → 7 → NULL
   Output: 1 → 3 → 5 → 7 → NULL

Problem 3: Find the Middle of a Linked List

Write a function that returns the middle node of a singly linked list. If there are two middle nodes (for lists with an even number of elements), return the second middle node.

Example: Input: 1 → 2 → 3 → 4 → 5 → NULL Output: The middle node is 3.
Input: 1 → 2 → 3 → 4 → NULL Output: The middle node is 3.

Problem 4: Separate Even and Odd Values

Write a function that takes a linked list as input and separates its even and odd values into two lists, maintaining the relative order of appearance in the original list.

Example: Input: 1 → 4 → 3 → 6 → 7 → NULL Output: Odd List: 1 → 3 → 7 → NULL Even List: 4 → 6 → NULL

Problem 5: Reverse the Linked List

Write a function to reverse a singly linked list. The function should modify the list in place and return the new head of the reversed list.

Example: Input: 1 → 2 → 3 → 4 → 5 → NULL Output: 5 → 4 → 3 → 2 → 1 → NULL

Final Task: Combine All Functions

Create a single C program that includes all the above functions in a meaningful way. Implement a menu-driven interface to let the user perform the following operations:

1. Create a linked list.

2. Create a cycle in the linked list.

3. Detect a cycle in the linked list.

4. Remove duplicates from a sorted linked list.

5. Find the middle of the linked list.

6. Separate even and odd values from the linked list.

7. Reverse the linked list.

Each of the above operations should be accessible through a menu, and the program should handle user inputs effectively.