******

WOLLO UNIVERSITY INSTITUTE OF TEACHNOLOGY CAMPUS

DATA STRUCTURE AND ALGORITHIM GROUP PROJECT

SOFTWARE ENGINERRING DEPARTMENT SECTION A

***GROUP ASSIGNMENT***

***GROUP MEMMBERS ID***

**1 AMIR HASSEN 0239/16**

**2 KALID MOHAMMED 1105/16**

**3 REHIMA YESUF HUSSEN 1624/16**

**4 ESKNDER MULUGETA 0720/16**

**5 MOHAMMED YESUF 4191/16**

**6 KERIYA ABDELLAH ALI 4170/16**

**Introduction**

This assignment focuses on the implementation of fundamental data structures: linked lists. Linked lists are a dynamic data structure that provides a flexible alternative to arrays, allowing for efficient insertion and deletion of elements. This project requires the development of four distinct types of linked lists in C++: singly linked lists, doubly linked lists, circular singly linked lists, and circular doubly linked lists. Each implementation will be encapsulated in its own file and will feature a menu-driven interface, enabling users to interact with the list and perform various operations. The goal is to gain a thorough understanding of linked list concepts, dynamic memory management, and the practical application of these data structures.

***Objectives***

Upon completion of this assignment, students will be able to:

**• Implement Linked List Variations:** Successfully create and implement four different types of linked lists: singly linked lists, doubly linked lists, circular singly linked lists, and circular doubly linked lists.

• **Master Dynamic Memory Management**: Utilize struct and dynamic memory allocation techniques (using new) to create and manage nodes within the linked lists.

**• Implement Core Linked List Operations:** Implement the fundamental operations for each linked list type, including:

**• Insertion**: Inserting nodes at the front, at the end, to the left of a specific value, and to the right of a specific value (where applicable).

**• Deletion**: Deleting nodes from the front, from the end, and a node with a specific value.

•  **Display:** Displaying the contents of the list in forward and backward directions (where applicable).

**• Apply Function Decomposition:** Employ function decomposition to create modular and reusable code for each operation (e.g., insert\_front(), delete\_end(), display\_list()).

•  **Develop Menu-Driven Interface:** Design and implement a user-friendly, menu-driven interface that allows users to interact with the linked lists and perform operations through clear prompts and messages.

**Understand Data Structure Trade-offs**: Gain an understanding of the advantages and disadvantages of each linked list type, including their performance characteristics and suitability for different applications.

**• Practice C++ Programming Skills:** Reinforce C++ programming skills, including the use of pointers, structures, dynamic memory allocation, and input/output operations.