INFORMATION
TECHNOLOGY
UNIVERSITY

Software Engineering Department - ITU
SE200L: Data Structures & Algorithms Lab

Course Instructor: Usama Bin Shakeel	**Dated: 18/09/2024**
Teaching Assistant: Zainab Bashir & Ryan Naveed	**Semester: Fall 2024**
Lab Engineer: Sadia Ijaz	**Batch: BSSE2023B**

Lab 5. Linked List Implementation and Manipulation

Name	**Roll number**	**Report (out of 35)**

Checked on: ______________________________

Signature: __________________________________

1.1 Objective

The objective of this lab is to practice problems related to Linked list implementation.

1.2 Equipment and Component

Component Description	Value	Quantity
Computer	Available in lab	1

1.3 Conduct of Lab

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

1.4 Theory and Background

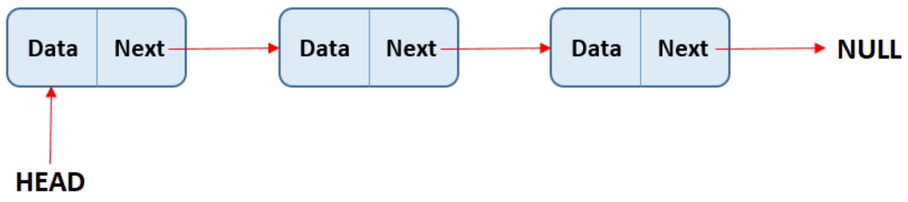
A **linked list** is a linear data structure where each element, called a node, points to the next node in the sequence using a pointer. Unlike arrays, linked lists do not need a contiguous block of memory and can dynamically grow or shrink during runtime.

Each node in a singly linked list consists of two parts:

1. **Data**: Stores the value (or data) of the node.
2. **Pointer**: Points to the next node in the sequence.

A **pointer** is a variable that stores the address of another variable. In linked lists, pointers are used to link one node to the next. The last node points to nullptr to indicate the end of the list.

The **linked list** is composed of nodes and typically has a head and tail pointer. The **head** points to the first node, while the **tail** points to the last node. Operations like insertion and deletion can be performed at various positions (head, tail, or any index).



Lab Tasks

Task 1

Implement the <u>Node</u> Class. The Node class represents a single node in the linked list. It will store an integer value and a pointer to the next node.

Data Members:

- int data: The data stored in the node.
- Node* next: Pointer to the next node in the list.

Member Functions to Implement:

- **Constructor**
- **Destructor**
- **void setNext(Node* val):** Sets the next pointer to the provided node.
- **Node* getNext( ):** Returns the next node pointer.
- **void setData(int data):** Sets the data of the node.
- **int getData( ):** Returns the data of the node.

Task 2

Implement the <u>List</u> Class. The List class manages the linked list and provides various functionalities to manipulate the nodes.

Data Members:

- **Node* head:** Pointer to the first node in the list.
- **Node* tail:** Pointer to the last node in the list.
- **int count:** The number of elements in the list.

Member Functions to Implement:

- **Constructor**: Initializes the linked list with head, tail, and count set to nullptr or 0.
- **Destructor**: Ensures all nodes are properly deleted when the list is destroyed.
- **bool isEmpty():** Returns true if the list is empty, otherwise false.
- **void append(int data):** Adds a new node with the given data at the end of the list.
- **void prepend(int data):** Adds a new node with the given data at the start of the list.
- **void insertAtIndex(int data, int index)**: Inserts a node with the given data at the specified index.
- **void deleteFromEnd():** Deletes the last node in the list.
- **void deleteFromStart():** Deletes the first node in the list.
- **void deleteFromIndex(int index):** Deletes the node at the specified index.
- **Node* getHead( ):** Returns head pointer

- **Node* getTail( ):** Returns tail pointer
- **void printList( ):**

Please read the following instructions carefully:

1. ***Do Not Modify test.cpp:*** *You are strictly prohibited from making any changes to the test.cpp file. This file is designed to test your implementation and any modifications will lead to the assignment being graded as zero.*
2. ***Class Definitions:*** *All class definitions and implementations must be provided solely within the files functions.h and functions.cpp. You are not allowed to create any additional files for your class definitions or implementations.*

Any deviation from these rules, including creating additional files or modifying the test.cpp file, will result in your assignment receiving a grade of zero.

Assessment Rubric for Lab

Performance metric	CLO	Able to complete the task over 80% (4-5)	Able to complete the task 50-80% (2-3)	Able to complete the task below 50% (0-1)	Marks
1. Realization of experiment	1	Executes without errors excellent user prompts, good use of symbols, spacing in output. The testing has been completed.	Executes without errors, user prompts are understandable,minimum use of symbols or spacing in output. Some testing has been completed.	Does not execute due to syntax errors, runtime errors, user prompts are misleading or non- existent. No testing has been completed.	
2. Conducting experiment	1	Able to make changes and answer all questions.	Partially able to make changes and few incorrect answers.	Unable to make changes and answer all questions.	
3. Computer use	2	Document submission timely.	Document submission late.	Document submission not done.	
4. Teamwork	3	Actively engages and cooperates with other group member(s) in an effective manner.	Cooperates with other group member(s) in a reasonable manner but conduct can be improved.	Distracts or discourages other group members from conducting the experiment	
5. Laboratory safety and disciplinary rules	3	Code comments are added and do help the reader to understand the code.	Code comments are added and do not help the reader to understand the code.	Code comments are not added.	
6. Data collection	3	Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap.	Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables.	Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy.	
7. Data analysis	4	Solution is efficient, easy to understand, and maintain.	A logical solution that is easy to follow but it is not the most efficient.	A difficult and inefficient solution.	
Total (out of 35):					