A REPORT
ON

CIRCULAR LINKED LIST

By

Name of the Students:	Registration No.
Y. HARDHIK	**AP24110011453**
P. DHATRI	**AP24110011459**
P. KIRANMAI	**AP24110011466**
B. SRI BHAVYA	**AP24110011431**
A. KEERTHAN	**AP24110011464**
SK. YUSUF	**AP24110011408**

Prepared in the partial fulfilment of the
Project Based Learning of Course CSE 201 – Coding Skills – 1

SRM UNIVERSITY, AP

DECEMBER 2025

CERTIFICATE

This is to certify that the project entitled **Circular Linked List** has been successfully completed by **Y. HARDHIK(AP24110011453)**, **P. DHATRI(AP24110011459), P. KIRANMAI(AP24110011466), B. SRI BHAVYA(AP24110011431), A. KEERTHAN(AP24110011464), SK. YUSUF(AP24110011408)** as a required academic component of the course *CSE 201: Coding Skills -1* during Semester 3 of Academic Year 2025-26 in the Department of Computer Science and Engineering at SRM University, AP. This work was executed under the guidance of the undersigned and is deemed to have successfully met all course requirements as of **09 December 2025.**

(Signature of the Course Instructor)

Professor Prakash,
Professor Naveen,
Department of Computer Science and Engineering,
SRM University, AP

ACKNOWLEDGEMENT

We, the project team, would like to express our sincere gratitude to all those who supported, guided, and encouraged us throughout the successful completion of our project.

First, we extend our heartfelt thanks to the **Hon’ble Vice Chancellor, Professor Ch. Satsih Kumar, of SRM University, AP**, for providing an excellent learning environment, advanced facilities, and continuous academic support that enabled us to carry out this project effectively.

We also convey our sincere appreciation to the **Dean, Professor CV Tomy, School of Engineering and Sciences**, for offering the necessary resources, academic guidance, and encouragement that greatly contributed to our learning and project development.

Our deepest gratitude goes to our **Faculty Mentor**, **Professor Prakash**, **Professor Naveen** whose continuous guidance, feedback, and motivation helped shape the direction of our project. Their patience, support, and expert advice were instrumental in overcoming challenges and completing this work successfully.

Finally, we wish to thank all faculty members, classmates, friends, and anyone who extended assistance, feedback, or encouragement during the course of our project. Their support has been truly meaningful and appreciated by our entire team.

ABSTRACT

The simulation of fundamental data structures plays a crucial role in helping students understand how algorithms behave internally. Among these structures, the *Circular Linked List* stands out due to its unique circular nature, where the last node connects back to the first, creating a continuous loop. This project presents an interactive and user-friendly **Circular Linked List Simulation with a Graphical User Interface (GUI)** developed using Python’s Tkinter library. The aim of the project is to transform theoretical concepts into an intuitive visual experience, enabling users to insert, delete, search, and traverse elements while directly observing how the circular structure updates in real time.

The simulation highlights each list operation by visually displaying the node sequence and its linkage back to the head, allowing learners to build a clear mental model of how circular linked lists function behind the scenes. This project not only strengthens conceptual understanding but also demonstrates how data structures can be enhanced with GUI-based visualization to improve accessibility and engagement. The application is lightweight, easy to operate, and suitable for both academic demonstrations and self-learning.

INTRODUCTION

Data structures form the foundation of modern computing, enabling efficient storage, retrieval, and manipulation of information. Among these structures, linked lists are particularly important due to their dynamic nature and flexibility. A *Circular Linked List* is a variation in which the final node connects back to the first, creating a continuous loop. This circular property makes the structure useful in scenarios such as round-robin scheduling, buffering, and applications requiring cyclic traversal.

For many learners, understanding how nodes link together in a circular manner can be challenging when relying solely on theoretical explanations. To bridge this gap, this project introduces a **GUI-based Circular Linked List Simulation** designed to visually demonstrate each operation step-by-step. By integrating an interactive interface using Python's Tkinter library, the system allows users to perform insertions, deletions, searches, and traversals while observing how the circular connections are updated in real time.

The goal of the project is not only to implement the circular linked list but also to present it in a way that enhances conceptual clarity. Visual cues, structured layout, and clear output feedback make the system approachable for students and helpful for educators demonstrating core data structure behavior. This simulation underscores how graphical tools can simplify complex concepts and foster a deeper understanding of algorithmic logic.

PROBLEM STATEMENT

Understanding the internal behavior of a circular linked list can be difficult for students, as traditional textbook explanations often fail to convey how node connections form a continuous loop. Without proper visualization, learners may struggle to interpret how insertions, deletions, and traversals affect the circular structure. This gap becomes even more apparent when teaching beginners who rely heavily on visual cues to grasp abstract concepts.

The lack of accessible and interactive tools for demonstrating circular linked list operations creates a need for a simple, intuitive simulation that allows users to observe dynamic structural changes. The challenge lies in developing a system that not only implements the underlying data structure correctly but also presents it through an engaging and easy-to-use graphical interface.

This project addresses these issues by creating a **GUI-based Circular Linked List Simulator** that visually represents node connections and provides real-time feedback for each operation. The system aims to make learning more intuitive, reduce conceptual confusion, and assist educators in effectively demonstrating circular linked list behavior.

OBJECTIVES

The primary objectives of this project are outlined as follows:

- **To develop a functional implementation of a Circular Linked List** that accurately supports core operations such as insertion, deletion, searching, and traversal.
- **To design a graphical user interface (GUI)** that visually represents the circular structure, enabling users to observe changes in real time.
- **To enhance conceptual understanding** by providing a dynamic visualization of how nodes connect, how the last node links back to the head, and how each operation affects the structure.
- **To create an interactive learning tool** suitable for students, educators, and beginners exploring data structures and algorithmic behavior.
- **To demonstrate the application of Python and Tkinter** for educational software development, showcasing how theoretical concepts can be translated into practical, visual simulations.
- **To offer a simple, accessible, and user-friendly environment** that encourages experimentation and supports classroom demonstrations.

PROJECT LAYOUT / SYSTEM ARCHITECTURE

The overall design of the Circular Linked List Simulation is structured in a modular and organized manner to ensure clarity, maintainability, and ease of understanding. The architecture integrates both backend data structure logic and a frontend graphical interface, allowing users to interact with the system intuitively while the underlying operations are executed efficiently.

The system is divided into two major layers:

1. **The Data Structure Logic Layer**, which handles the implementation of the circular linked list.
2. **The Graphical User Interface (GUI) Layer**, which communicates with the user, triggers list operations, and displays real-time results.

Together, these layers create a seamless environment where algorithmic behavior becomes visible and interactive.

1. Data Structure Logic Layer

At the core of the system lies the implementation of the **Circular Linked List**, which is built using a `Node` class and a `CircularLinkedList` class.

Node Structure

Each node contains two key attributes:

- `data` – the value stored in the node
- `next` – a pointer linking to the next node, forming the circular structure

The last node always points back to the head, closing the loop and ensuring continuous traversal.

Circular LinkedList Class

This class manages all the primary operations, including:

- **Insertion at front**
- **Insertion at end**
- **Deletion of a specific value**
- **Searching for a value in the list**
- **Traversal and display of the entire circular structure**

Special care is taken to maintain the circular nature during each operation, ensuring pointer adjustments preserve the loop even after modifications.

The logic layer is intentionally separated from the GUI to allow modular design, making it easier to test and reuse independently.

2. Graphical User Interface Layer

The GUI is developed using Python’s **Tkinter library**, providing a clean and interactive window-based environment. Its primary purpose is to:

- Accept user input
- Trigger backend operations
- Display the updated circular linked list
- Provide visual confirmation of each action

Input Section

A simple text entry widget allows the user to type values that can be inserted, searched, or deleted.

Action Buttons

Buttons are provided for core operations such as:

- Insert Front
- Insert End
- Delete Value
- Search Value
- Reset List

Each button is linked to a specific function that communicates with the circular linked list backend.

Display Panel

A text box or label is used to visually present the current sequence of nodes.
This allows users to immediately observe:

- How nodes are connected
- The circular nature of the structure
- The effect of each operation

The GUI enhances accessibility, making the system approachable for beginners and visually intuitive for learners.

3. Interaction Flow

The system follows a clear and direct interaction flow:

1. **User enters a value** in the input field.
2. **User selects an operation** from the available buttons.
3. The GUI calls the **corresponding method** in the CircularLinkedList class.
4. The backend executes the operation and adjusts the pointers accordingly.
5. The result is returned to the GUI.
6. The updated list is **visually displayed**, showing the new circular structure.

This flow ensures that complex pointer manipulations happen behind the scenes while the user experiences a smooth and easy-to-understand interface.

4. Design Considerations

Several principles guided the system design:

- **Modularity:** Logic and interface are separated into independent layers.
- **Simplicity:** Operations are presented clearly to avoid confusion for new learners.
- **Real-time Visual Feedback:** Each change is immediately visible, enhancing conceptual clarity.
- **Error Handling:** The system manages invalid inputs, empty list operations, and non-existent values gracefully.

This architecture makes the Circular Linked List Simulation not only functionally complete but also pedagogically effective. It mirrors real-world software design practices while keeping the learning curve gentle for students exploring data structures.

MODULES EXPLANATION

The Circular Linked List Simulation has been structured into multiple modules to ensure clarity, maintainability, and logical separation of responsibilities. Each module contributes to the overall functionality by handling a specific aspect of the system. The following sections describe each module in detail.

1. Node Module

The **Node Module** represents the fundamental building block of the circular linked list.
Each node consists of:

- **Data Field:** Stores the actual value of the node.
- **Next Pointer:** Refers to the next node in the sequence and is responsible for maintaining the circular structure.

The final node's `next` pointer always points back to the head, forming a continuous loop. This module abstracts individual node creation and ensures each new node correctly integrates into the existing structure.

2. Circular Linked List Operations Module

This module forms the core of the system, as it implements all the essential operations of a circular linked list:

a. Insertion Operations

- **Insert at Front:** Adds a new node at the beginning of the list while updating the last node's pointer to maintain circularity.
- **Insert at End:** Appends a new node to the end and ensures the new node loops back to the head node.

b. Deletion Operation

- **Delete by Value:** Searches for a specific value and removes the corresponding node, adjusting the pointers to preserve the loop.

c. Searching

- Identifies whether a given value exists within the list.
- Provides a simple boolean response that is used by the GUI for alarms or notifications.

d. Traversal and Display

- Iterates through the list until it loops back to the head, collecting node values for representation.
- Returns a formatted string that visually illustrates the continuous structure.

This module handles all pointer manipulation, loop maintenance, and edge-case management (such as deletion from an empty list or removing the head node).

3. GUI Interaction Module

Built using Python’s Tkinter library, this module provides the interactive graphical interface that allows users to input values and perform operations visually.

a. Input Handling

An entry widget collects numerical values from the user. This module also includes input validation to prevent invalid or empty submissions.

b. Buttons and Commands

A set of buttons triggers all available operations:

- **Insert Front**
- **Insert End**
- **Delete Value**
- **Search Value**
- **Reset List**

Each button is mapped to a corresponding function that communicates with the Circular Linked List Operations Module.

c. Output Display Panel

A text box or label is used to present the current state of the circular linked list:

- Displays the sequence of nodes
- Highlights the continuous nature with a pointer to the head
- Updates after every operation, providing real-time feedback

This module ensures a user-friendly and intuitive experience by linking interactive controls with backend operations.

4. System Control & Flow Module

This module coordinates communication between the GUI and the backend logic. When a user interacts with the interface:

1. The input is validated.
2. The corresponding linked list operation is called.
3. The updated list structure is returned.
4. The GUI refreshes the output display.

This module ensures seamless integration between user actions and data structure behavior. It acts as the bridge that synchronizes visual representation with algorithmic operations.

5. Error Handling and Validation Module

To ensure smooth execution and avoid runtime issues, this module manages:

- **Invalid inputs** (non-numeric entries)
- **Operations on an empty list**
- **Deletion of non-existent values**

Custom notifications through message boxes inform users of errors without interrupting program execution.

These modules collectively form a cohesive simulation tool that is efficient, visually intuitive, and highly suitable for academic purposes. The separation of logic, interface, and control simplifies understanding and enhances long-term maintainability of the system.

FEATURES IMPLEMENTED

The Circular Linked List Simulation incorporates a comprehensive set of features designed to enhance the learning experience and provide an intuitive understanding of the data structure's behavior. The combination of backend logic and graphical user interface ensures that each operation is both functional and visually interpretable. The major features implemented in the system are detailed below.

1. Interactive Graphical User Interface (GUI)

A user-friendly GUI built using Python's Tkinter library allows seamless interaction with the circular linked list.
The interface includes:

- A clear input field for entering node values
- Dedicated buttons for each operation
- Instant visual feedback through an output display panel
- Error messages for invalid or out-of-range operations

The GUI transforms theoretical operations into real-time visual actions, making the learning process more engaging.

2. Dynamic Circular Linked List Construction

The system fully supports the creation and dynamic modification of a circular linked list.
Key capabilities include:

a. Insert at Front

Adds a new node to the beginning of the list while ensuring that the last node correctly points to the new head.

b. Insert at End

Appends a node at the end and maintains the circular reference back to the head node.

These operations demonstrate how a circular list maintains continuity even during structural changes.

3. Node Deletion by Value

Users can remove a node containing a specific value.
The program handles:

- Deletion of the head node
- Deletion of middle and end nodes
- Cases where the value does not exist

Pointer adjustments are performed carefully to preserve the circular property after deletion, highlighting real-world linked list memory behavior.

4. Search Operation

The system provides a feature to search for a specific value in the list.
It:

- Traverses the circular structure safely
- Informs the user whether the value exists
- Enhances conceptual understanding of list traversal in looped structures

This is especially important for learners recognizing the difference between linear and circular traversal patterns.

5. Real-Time Visual Display of Circular Structure

One of the most valuable features is the real-time display of the list after every operation.
The visual output:

- Shows all node values in sequence
- Indicates the circular link back to the head
- Updates automatically after every action

This helps users clearly visualize how nodes are linked and how operations affect the entire structure.

6. Reset Functionality

A reset option allows users to clear the entire list and rebuild from scratch.
This enables rapid experimentation and reinforces conceptual understanding through repeated practice.

7. Robust Error Handling

To ensure smooth operation and prevent unexpected behavior, the system includes:

- Input validation to prevent non-numeric entries
- Proper handling of empty list conditions
- Informative alerts when attempting invalid operations
- Controlled responses for deletion and search failures

These safeguards improve user experience and prevent program crashes.

8. Modular and Maintainable Code Design

The system is implemented with a modular structure separating:

- Node logic
- Circular linked list operations
- GUI interaction
- Input/output management

This ensures:

- Clean readability
- Easy debugging
- Simple future enhancements

It also follows standard software development practices taught in academic courses.

These features collectively make the Circular Linked List Simulation an effective educational tool, allowing users to learn through interaction, visualization, and structured experimentation.

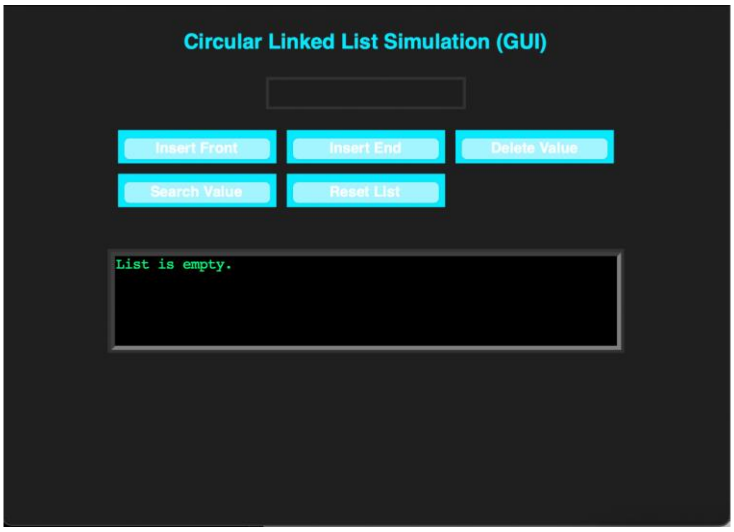
SCREENSHOTS

This section presents the visual outputs of the Circular Linked List Simulation executed through the GUI developed using Python's Tkinter library. Each screenshot highlights a specific operation performed on the circular linked list, allowing clear observation of how the structure updates in real time.
(Note: Replace each placeholder with your actual screenshots before submission.)

1. Application Startup Window

Description:
This screenshot displays the initial user interface of the Circular Linked List Simulator. It includes the input field, operation buttons, and the output display area.

Placeholder:

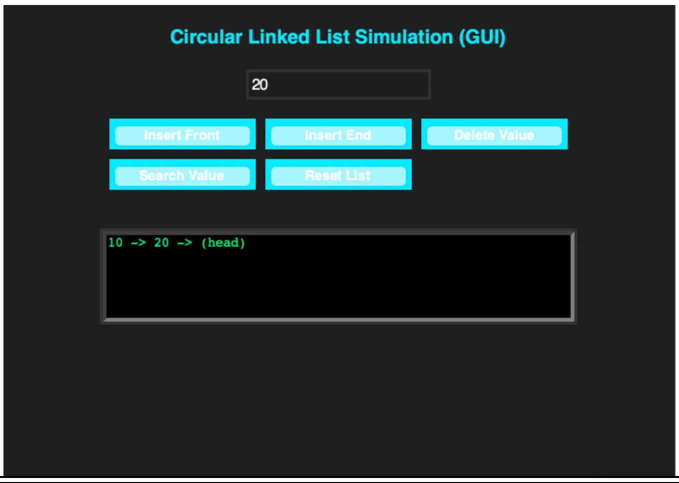


```
List is empty.
```

2. Inserting Nodes into the Circular Linked List

Description:
This screenshot shows the list after inserting nodes using the **Insert Front** and **Insert End** options. The output area reflects the updated sequence and circular linkage.

Placeholder:

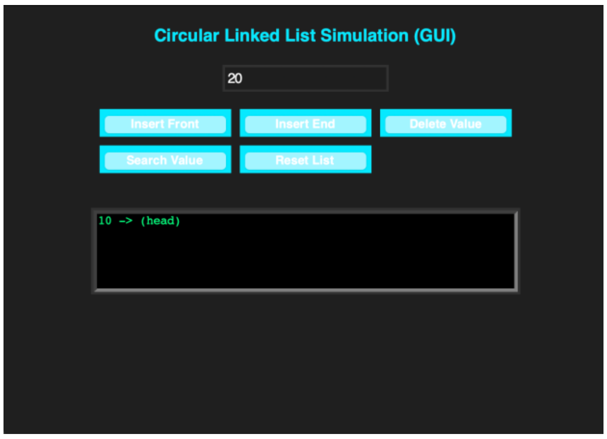


```
10 -> 20 -> (head)
```

3. Deleting a Node

Description:
This screenshot demonstrates the deletion of a specific value from the circular linked list.
It highlights how the structure remains intact even after removal.

Placeholder:

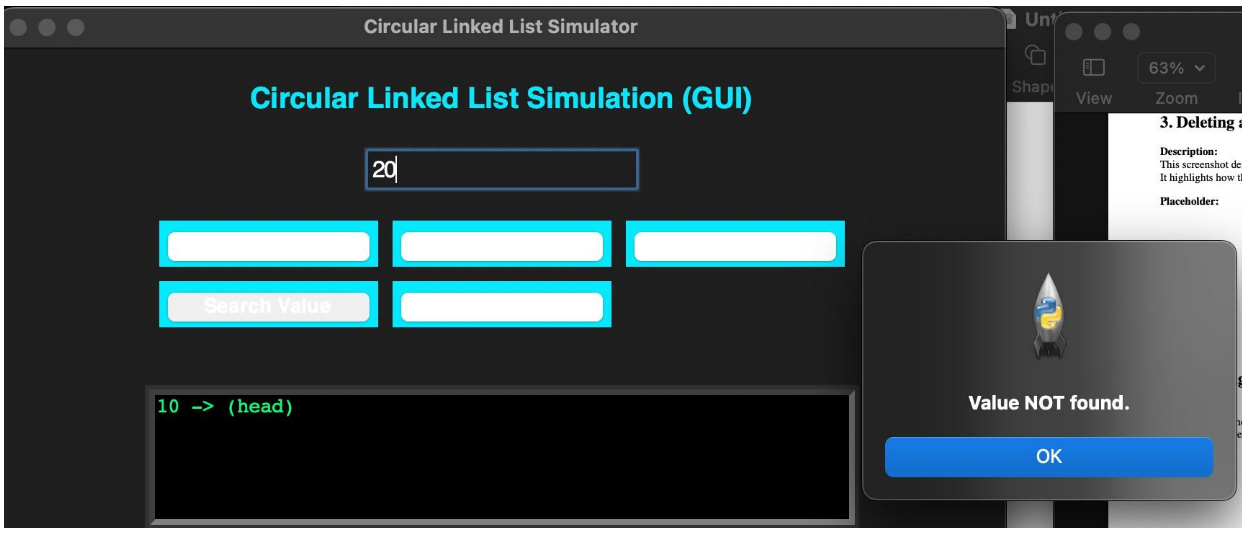


```
10 -> (head)
```

4. Searching for a Value

Description:
This screenshot shows the message box result when the user performs a search operation.
It confirms whether the value exists in the circular linked list.

Placeholder:


Circular Linked List Simulator
Circular Linked List Simulation (GUI)
20
Search Value
OK

```
10 -> (head)
```

Value NOT found.

CONCLUSION

The Circular Linked List Simulation developed in this project successfully demonstrates how a fundamental data structure can be understood more effectively through graphical visualization. By integrating Python's Tkinter library with a fully functional circular linked list implementation, the system provides a clear and interactive way to observe how insertions, deletions, searches, and traversals affect the structure. The simulation makes abstract concepts tangible by presenting real-time updates and feedback, which enhances comprehension and supports learning in a practical manner.

The project also highlights the importance of separating core logic from user interface components, contributing to a cleaner and more modular design. This approach not only simplifies debugging and maintenance but also reflects real-world software development practices. Furthermore, the GUI-based environment makes the system approachable for beginners, educators, and students, offering an intuitive and engaging learning experience.

Overall, the objectives of the project have been fully met. The simulation reinforces theoretical understanding, encourages experimentation, and provides a visually driven platform for exploring circular linked lists. It serves as a strong foundation for future enhancements and demonstrates the value of combining data structures with interactive visualization tools.

FUTURE ENHANCEMENTS

While the current version of the Circular Linked List Simulation successfully demonstrates all core operations through a graphical interface, there remain several opportunities to further improve the system’s functionality, usability, and educational value. The following enhancements may be considered for future development:

1. Advanced Visualization of Node Connections

At present, the structure is displayed textually. A more sophisticated visual representation could illustrate each node as a graphical object connected by arrows. Animating insertions, deletions, and pointer transitions would provide learners with a clearer understanding of how node links are manipulated internally.

2. Support for Additional Operations

The simulator can be expanded to include more advanced circular linked list operations such as:

- Splitting the circular list into two halves
- Josephus problem simulation
- Sorted insertion
- Reversing the circular list

These operations would deepen the user’s understanding of the flexibility and applicability of circular linked lists.

3. Integration of Doubly Circular Linked Lists

Extending the system to support **doubly circular linked lists** would introduce users to a broader range of pointer structures. This enhancement would highlight differences in traversal, deletion efficiency, and the use of bidirectional navigation.

4. Enhanced User Interface Design

The GUI can be improved through:

- Color themes and customizable layouts
- Improved button styling
- Tooltips explaining each operation
- A clearer output area with formatting and dynamic highlighting.

5. Step-by-Step Execution Mode

Introducing a slow-motion or step-through mode would allow users to observe the exact sequence of pointer updates during operations. This feature would be especially beneficial for students who struggle to visualize how lists change internally.

6. Export and Import Functionality

Allowing users to save their current list state to a file and reload it later would make the system more practical for demonstrations and assessments. This enhancement also provides continuity across sessions.

7. Web-Based Implementation

A browser-compatible version using web technologies such as HTML, CSS, and JavaScript could increase accessibility. A web-based simulator would eliminate installation requirements and allow the tool to be shared easily with larger groups of students.

8. Integration with Educational Platforms

The simulation could be embedded within learning management systems or used as part of interactive lab assignments. This expansion would support classroom use and structured curriculum integration.

These enhancements would significantly expand the functionality and pedagogical impact of the Circular Linked List Simulation, transforming it from a simple demonstration into a comprehensive learning tool capable of supporting deeper exploration of data structures.

REFERENCES

1. Lafore, R. (2017). *Data Structures and Algorithms in C++*. Sams Publishing.
2. Weiss, M. A. (2014). *Data Structures and Algorithm Analysis in C++*. Pearson Education.
3. Sedgewick, R., & Wayne, K. (2011). *Algorithms (4th Edition)*. Addison-Wesley.
4. Stroustrup, B. (2013). *The C++ Programming Language*. Addison-Wesley Professional.
5. Python Software Foundation. “Tkinter Documentation.” Retrieved from https://docs.python.org
6. GeeksforGeeks. “Circular Linked List – Introduction and Operations.” Retrieved from https://www.geeksforgeeks.org
7. TutorialsPoint. “Python Tkinter – GUI Programming.” Retrieved from https://www.tutorialspoint.com
8. MIT OpenCourseWare. “Introduction to Algorithms and Data Structures.” Massachusetts Institute of Technology.
9. University Classroom Notes and Lecture Materials on Circular Linked Lists (2024).
10. Tkinter Official Wiki. “Graphical User Interface Development in Python.” Retrieved from https://wiki.python.org/moin/TkInter