UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 6

Singly Linked Lists

Submitted by:
Adoracion, Jerick Dave D.

Instructor:
Engr. Maria Rizette H. Sayo

August 23, 2025

I. Objectives

Introduction

A linked list is an organization of a list where each item in the list is in a separate node. Linked lists look like the links in a chain. Each link is attached to the next link by a reference that points to the next link in the chain. When working with a linked list, each link in the chain is called a Node. Each node consists of two pieces of information, an item, which is the data associated with the node, and a link to the next node in the linked list, often called next.

This laboratory activity aims to implement the principles and techniques in:

- Writing algorithms using Linked list
- Writing a python program that will perform the common operations in a singly linked list

II. Methods

- Write a Python program to create a singly linked list of prime numbers less than 20. By iterating through the list, display all the prime numbers, the head, and the tail of the list. (using Google Colab)
- Save your source codes to GitHub

III. Results

```
class LinkedListNode:
    def __init__(self, prime_num, nextNode=None):
        self.prime_num = prime_num
        self.nextNode = nextNode
```

Figure 1: Source Code (Node Class)

The LinkedListNode class creates a node that holds a prime number and a pointer to the next node, forming the basic unit of the list.

```
#adding a limit function since the numbers to be used are only less than 20
class LinkedList:
    def __init__(self, limit=20, head=None, tail=None):
        self.limit = limit
        self.head = head
        self.tail = tail
```

Figure 2: Source Code (LinkedList Class)

This LinkedList class manages the list with a limit (default 20), a head pointing to the first node, and a tail pointing to the last node.

```
def insert(self, prime_num):
    if prime_num > self.limit:
        print(f"\n{prime_num} exceeds the limit ({self.limit}) and will not be inserted.")
        return

    node = LinkedListNode(prime_num)
    if self.head is None:  # If list is empty
        self.head = node
        self.tail = node
    else:
        self.tail.nextNode = node
        self.tail = node
```

Figure 3: Source Code (Insert Method)

The insert method adds a new node if the number is within the limit. If the list is empty, the node becomes both head and tail; otherwise, it's added at the end.

```
def display(self):
    current = self.head
    while current is not None:
        print(current.prime_num, "->", end= " ")
        current = current.nextNode
    print(None)
```

Figure 4: Source Code (Display Method)

The display method traverses from the head and prints all nodes in sequence until reaching None.

```
def show_head(self):
    if self.head is not None:
        print(f"Head: {self.head.prime_num}")
    else:
        print("The list is empty.")

def show_tail(self):
    if self.tail is not None:
        print(f"Tail: {self.tail.prime_num}")
    else:
        print("The list is empty.")
```

Figure 5: Source Code (Show Head and Show Tail)

These methods display the first and last elements of the list or indicate if the list is empty.

```
ll = LinkedList(limit=20)

ll.show_head()  # to show head is empty
primes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]  # I included 23, and 29 to test the limit

for p in primes:
    ll.insert(p)
    ll.display()

ll.show_head()
ll.show_tail()
```

Figure 6: Source Code (Testing the Program)

The program inserts prime numbers below 20 into the list, rejects values above the limit, and displays the final sequence from head (2) to tail (19).

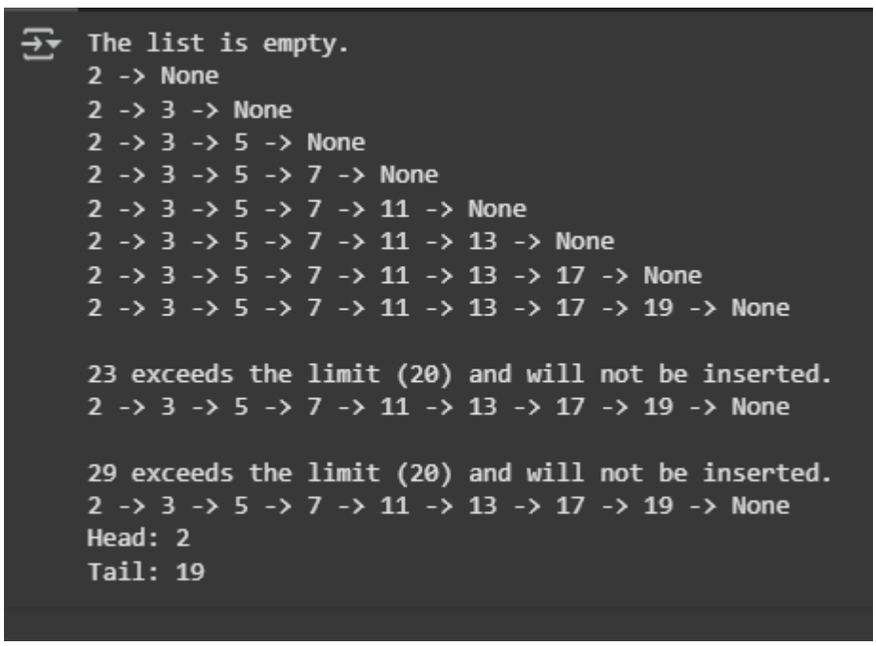

Figure 7: Output

The output shows the linked list being built step by step as each prime number is inserted. At first, the list is empty, but as numbers are added, it grows from `2 -> None` to `2 -> 3 -> 5 -> ... -> 19 -> None`. When 23 and 29 are inserted, the program rejects them since they exceed the set limit of 20. Finally, the head of the list is confirmed as 2 and the tail as 19, showing that the list correctly stores only prime numbers below 20.

IV. Conclusion

As a beginner in Computer Engineering, this activity on singly linked lists helped me understand how data can be stored and connected using nodes. At first, it was challenging to grasp how the head, tail, and pointers work together, but by coding and testing the program, I was able to see how each prime number was linked step by step. I also learned the importance of setting conditions, like limiting the values to numbers below 20, to ensure the program works correctly. Overall, this laboratory exercise gave me a strong foundation in linked list operations and boosted my confidence in handling more complex data structures in the future.

References

[1] Co Arthur O.. “University of Caloocan City Computer Engineering Department Honor Code,” UCC-CpE Departmental Policies, 2020.

[2] A. O. Co, “University of Caloocan City Computer Engineering Department Honor Code,” *UCC-CpE Departmental Policies*, 2020.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA: MIT Press, 2009.

[4] M. Goodrich, R. Tamassia, and M. Goldwasser, *Data Structures and Algorithms in Python*. Hoboken, NJ: Wiley, 2013.

[5] J. Zelle, *Python Programming: An Introduction to Computer Science*, 3rd ed. Franklin, Beedle & Associates Inc., 2016.