UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 6

Singly Linked Lists

Submitted by:
Enverzo, Kyle Andrey 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

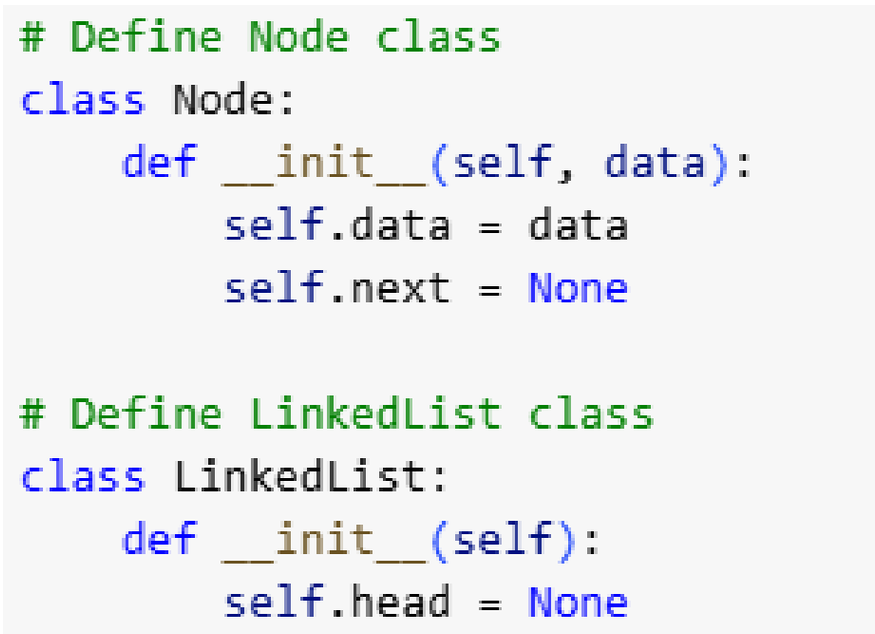
II. Methods

- 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

Present the visualized procedures done. Also present the results with corresponding data visualizations such as graphs, charts, tables, or image . Please provide insights, commentaries, or explanations regarding the data. If an explanation requires the support of literature such as academic journals, books, magazines, reports, or web articles please cite and reference them using the IEEE format.

Please take note of the styles on the style ribbon as these would serve as the style format of this laboratory report. The body style is Times New Roman size 12, line spacing: 1.5. Body text should be in Justified alignment, while captions should be center-aligned. Images should be readable and include captions. Please refer to the sample below:

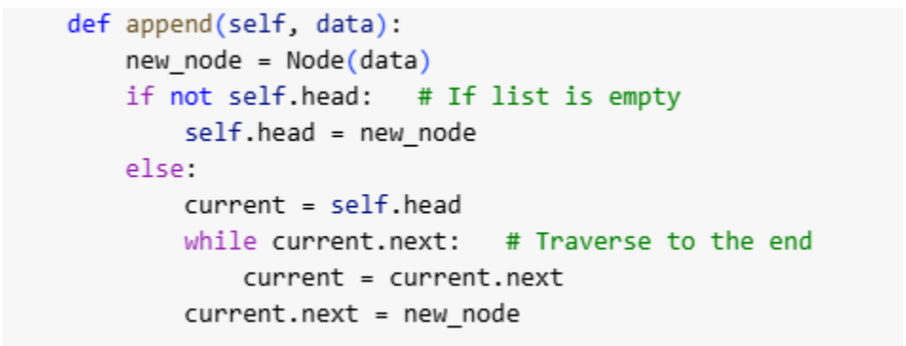


```
# Define Node class
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

# Define LinkedList class
class LinkedList:
    def __init__(self):
        self.head = None
```

Figure 1 Screenshot of program

Figure 1 defines the Node class, where each node stores some data and a pointer (next) to the next node in the linked list, initially set to None. It also initializes the LinkedList class with a head pointer that starts as None, meaning the list is empty at first.



```
def append(self, data):
    new_node = Node(data)
    if not self.head:   # If list is empty
        self.head = new_node
    else:
        current = self.head
        while current.next:   # Traverse to the end
            current = current.next
        current.next = new_node
```

Figure 2 Screenshot of program

In Figure 2, append method adds a new node at the end of the linked list by traversing from the head to the tail and linking the new node there; if the list is empty, the new node becomes the head.

```
def display(self):
    current = self.head
    while current:
        print(current.data, end=" , ")
        current = current.next
    print("None")
```

Figure 3 Screenshot of program

This Figure 3 display method prints all elements in the linked list by starting from the head and moving through each node until it reaches the end, marking the termination with None.

```
def get_head(self):
    return self.head.data if self.head else None
```

Figure 4 Screenshot of program

In This Figure 4, method returns the data of the head node if the list isn't empty; otherwise, it returns None.

```
def get_tail(self):
    current = self.head
    if not current:
        return None
    while current.next:
        current = current.next
    return current.data
```

Figure 5 Screenshot of program

Figure 5 get tail method traverses from the head to the last node (tail) and returns its data, or None if the list is empty.

```python
# Function to check prime
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num**0.5)+1):
        if num % i == 0:
            return False
    return True

# Create linked list of primes < 20
ll = LinkedList()
for i in range(20):
    if is_prime(i):
        ll.append(i)

# Display results
print("Prime Numbers in Linked List:")
ll.display()

print("Head of list:", ll.get_head())
print("Tail of list:", ll.get_tail())
```

Figure 6 Screenshot of program

Figure 6, is_prime function checks whether a number is prime by testing divisibility up to its square root, returning True for primes and False otherwise. Using this function, the program creates a linked list (ll) that stores all prime numbers less than 20 by appending them one by one. Finally, the program displays the prime numbers in the linked list along with the first element (head) and the last element (tail).

IV. Conclusion

The program successfully demonstrates how to use a singly linked list to store and manage data—in this case, prime numbers less than 20. By defining the Node and LinkedList classes, it shows how nodes are connected and traversed, while methods like append, display, get_head, and get_tail provide basic linked list operations. The integration of the is_prime function highlights how custom logic can be combined with data structures to solve specific problems. Overall, the code provides a clear example of applying linked lists in Python to organize and display prime numbers efficiently.

References

[1] M. A. Weiss, *Data Structures and Algorithm Analysis in C++*, 4th ed. Boston, MA, USA: Pearson, 2013.

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

[3] N. Wirth, *Algorithms + Data Structures = Programs*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1976.

[4] Python Software Foundation, "Python documentation: Classes," *Python.org*. [Online]. Available: https://docs.python.org/3/tutorial/classes.html. [Accessed: Aug. 23, 2025].

[5] G. van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA, USA: CreateSpace, 2009.