UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 6

Singly Linked Lists

Submitted by:
Delinia, Filjohn B.

Instructor:
Engr. Maria Rizette H. Sayo

08, 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.

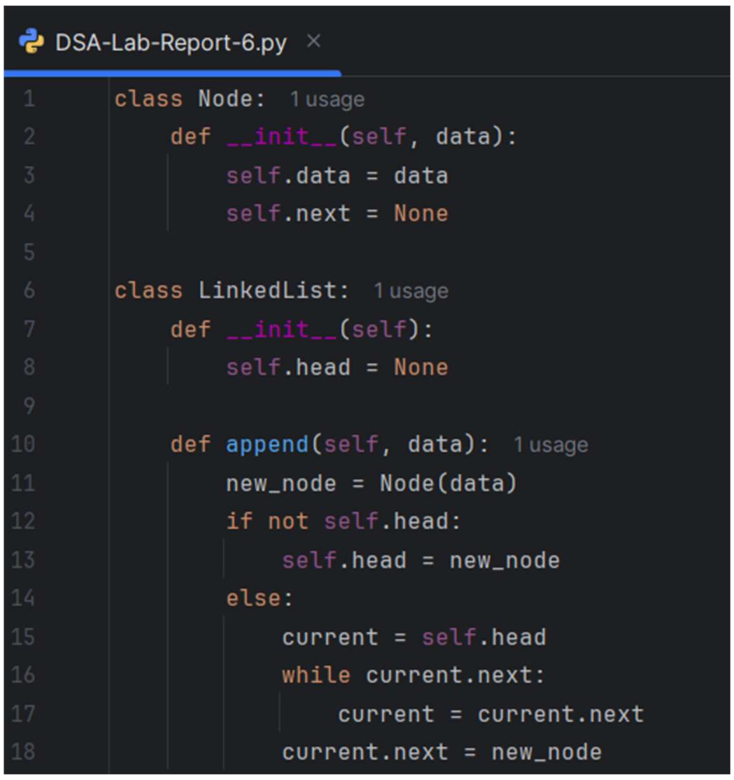
This laboratory activity aims to implement the principles and techniques in:

- 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

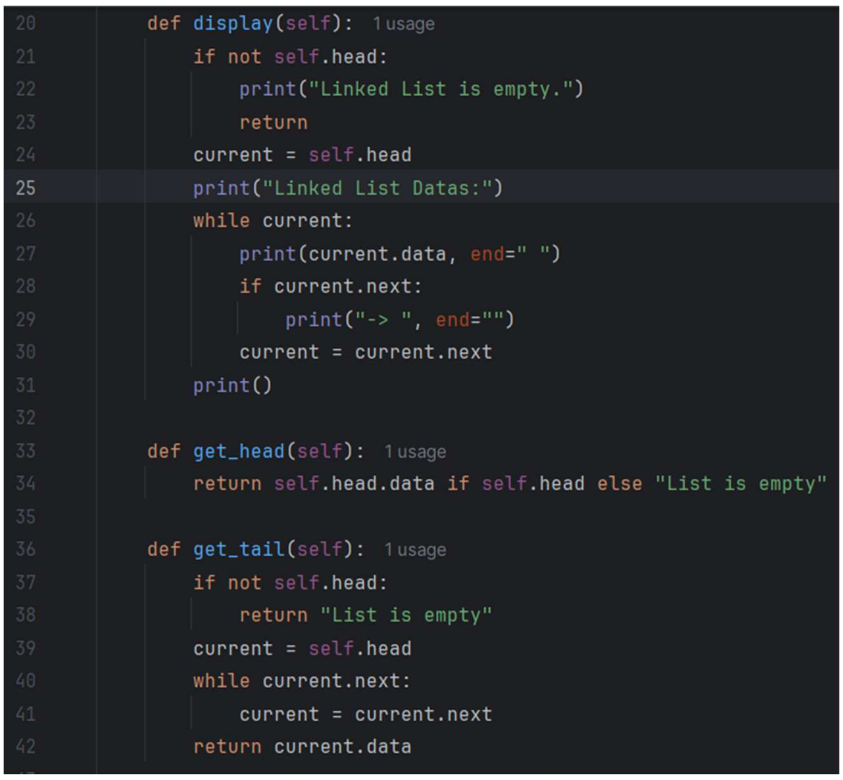


```python
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

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

    def append(self, data):
        new_node = Node(data)
        if not self.head:
            self.head = new_node
        else:
            current = self.head
            while current.next:
                current = current.next
            current.next = new_node
```

Figure 1. Source Code

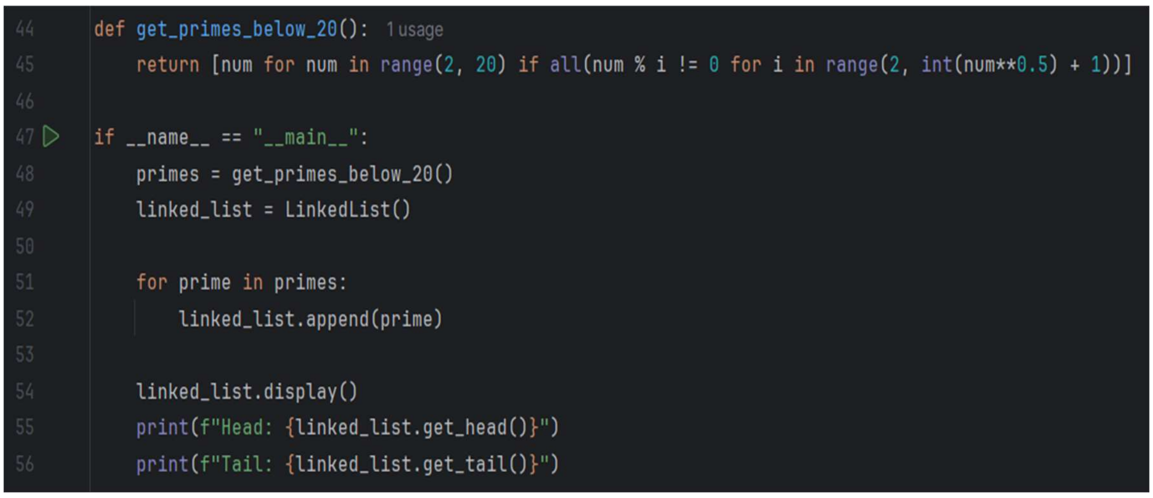


```
    def display(self):
        if not self.head:
            print("Linked List is empty.")
            return
        current = self.head
        print("Linked List Datas:")
        while current:
            print(current.data, end=" ")
            if current.next:
                print("-> ", end="")
            current = current.next
        print()

    def get_head(self):
        return self.head.data if self.head else "List is empty"

    def get_tail(self):
        if not self.head:
            return "List is empty"
        current = self.head
        while current.next:
            current = current.next
        return current.data
```

Figure 2. Source Code



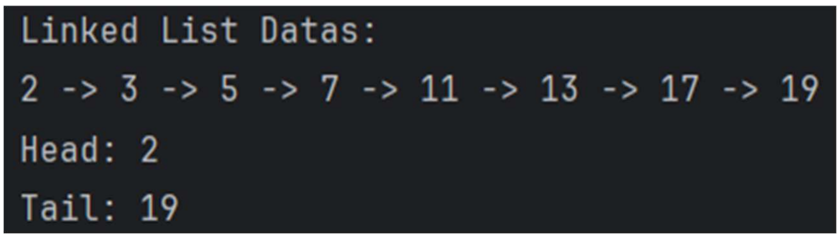
```
def get_primes_below_20():
    return [num for num in range(2, 20) if all(num % i != 0 for i in range(2, int(num**0.5) + 1))]

if __name__ == "__main__":
    primes = get_primes_below_20()
    linked_list = LinkedList()

    for prime in primes:
        linked_list.append(prime)

    linked_list.display()
    print(f"Head: {linked_list.get_head()}")
    print(f"Tail: {linked_list.get_tail()}")
```

Figure 3. Source Code



```
Linked List Datas:
2 -> 3 -> 5 -> 7 -> 11 -> 13 -> 17 -> 19
Head: 2
Tail: 19
```

Figure 4. Output

IV. Conclusion

After exploring how linked lists work and how to generate prime numbers, I now understand how data can be organized and connected efficiently. Building a singly linked list of primes helped me apply object-oriented concepts and sharpen my logic. This gave me a clearer grasp of both data structures and algorithm design.

References

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