UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 6

Singly Linked Lists

Submitted by:
Villanueva, Bryan O.

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

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

    def append(self, data):
        new_node = Node(data)
        if self.head is None:
            self.head = new_node
        else:
            current = self.head
            while current.next:
                current = current.next
            current.next = new_node

    def display(self):
        current = self.head
        while current:
            print(current.data, end=" -> ")
            current = current.next
        print("Null")

    def get_head_tail(self):
        if self.head is None:
            return None, None
        head_value = self.head.data
        current = self.head
        while current.next:
            current = current.next
        tail_value = current.data
        return head_value, tail_value
```

```
    def get_head_tail(self):
        if self.head is None:
            return None, None
        head_value = self.head.data
        current = self.head
        while current.next:
            current = current.next
        tail_value = current.data
        return head_value, tail_value

    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

linked_list = SinglyLinkedList()
for i in range(20):
    if is_prime(i):
        linked_list.append(i)


print("Prime numbers in the linked list:")
linked_list.display()

head, tail = linked_list.get_head_tail()
print(f"Head of the list: {head}")
print(f"Tail of the list: {tail}")
```

```
Prime numbers in the linked list:
2 -> 3 -> 5 -> 7 -> 11 -> 13 -> 17 -> 19 -> Null
Head of the list: 2
Tail of the list: 19
```

Figure 1 Screenshot of the program

The code shows how linked lists work in Python by building a list of prime numbers less than 20. I learned that a linked list is made of nodes, where each node has data and a link to the next node. This is different from a normal list because linked lists use pointers to connect the data, making it easier to add or remove elements without changing the whole structure. I also saw how a simple function can check if numbers are prime, and how looping through the list lets us see each number one by one. This activity helped me understand both prime number checking and basic data structures better.

IV. Conclusion

In this laboratory, I learned how to create a singly linked list in Python and use it to store prime numbers less than 20. I also practiced writing a function to check for prime numbers and understood how to add nodes to the list, display them, and find the head and tail. This helped me understand how linked lists work and how data can be connected and accessed step by step.

References

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

[2]“W3Schools.com,” *W3schools.com*, 2025. https://www.w3schools.com/python/python_dsa_linkedlists.asp (accessed Aug. 23, 2025).