UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 6

Singly Linked Lists

Submitted by:
Balaoro, Judge Wayne B.

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

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

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

  def insert(self, prime_num):
    node = LinkedListNode(prime_num)
    if self.head is None:
          self.head = node
          self.tail = node
    else:
      self.tail.nextNode = node
      self.tail = node

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

```python
ll = LinkedList()

ll.show_head() #to show the head is empty
ll.insert(2)
ll.display()

ll.insert(3)
ll.display()
ll.show_tail() #to show the current tail

ll.insert(5)
ll.display()

ll.insert(7)
ll.display()

ll.insert(11)
ll.display()

ll.insert(13)
ll.display()

ll.insert(17)
ll.display()

ll.insert(19)
ll.display()

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

```
The list is empty.
2 ->  None
2 ->  3 ->  None
Tail: 3
2 ->  3 ->  5 ->  None
2 ->  3 ->  5 ->  7 ->  None
2 ->  3 ->  5 ->  7 ->  11 ->  None
2 ->  3 ->  5 ->  7 ->  11 ->  13 ->  None
2 ->  3 ->  5 ->  7 ->  11 ->  13 ->  17 ->  None
2 ->  3 ->  5 ->  7 ->  11 ->  13 ->  17 ->  19 ->  None
Head: 2
Tail: 19
```

Figure 1. Source Code

In this laboratory activity, I am to program a singly linked list of prime numbers less than 20. First, I created a LinkedListNode class to represent the nodes in the singly linked list and I put a parameter of prime_num and the nextNode which I defaulted to empty since if we are to start with a new node the nextNode will be empty. Next, I created a LinkedList class to organize the insert method, display method, show head and tail method. On the constructor I set the head and tail to None since as said earlier if we are to start on a new node the head and tail are still empty until we add an argument for LinkedListNode. On the insert method I set the one that takes the argument of prime numbers to Node and I make an if-else statement that if the head is empty, that inserted node will become the head and tail as its just the first node and else when there's a tail, we will connect that tail into the nextNode and that nextNode will become the new tail. On the display method. I set the head into the current as it is the first Node. and while the pointer points to a newNode, it will print the current prime number with the string of "->" to make it more readable that it points into a new node and I used the end function so that it won't print into the new line and make it more confusing. On the show_head and show_tail method, I

used the if-else statement to check if the head or tail is not empty, it will print the head or tail and if it's empty it will print that the list is empty.

I set the LinkedList() to ll so that I won't have to type a long word and I could just type a two letter, I used the show_head method on the start so that I could show that the list is empty and also after used the insert method to enter the first and second prime number which is 2 and 3 I used the show_tail method to show that the current tail is 3. I did the display method everytime I inserted a prime number to show that it works as intended and used the show_head and show_tail method at the end to show the head and tail which is 2 and 19.

IV. Conclusion

In this laboratory activity. I was able to learn about linked lists and singly linked lists specifically. I learned the difference between array and linked lists while working on this activity and that the node contains two objects which are the value and null. The null points to the next node which will be the current tail and so on. If we are starting on a first node, that first node will become the current head and tail and after we enter the new node the current new node will become the current tail. Thanks to this activity I was able to apply things I learned about singly linked list, and create many methods inside the linked list.

References

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

[2] *Bot verification*. (n.d.). https://pythonguides.com/print-in-the-same-line-in-python/

[3] GeeksforGeeks. (2025, July 23). *Singly linked list in Python*. GeeksforGeeks. https://www.geeksforgeeks.org/python/singly-linked-list-in-python/

[4] Anthony Sistilli. (2018, April 14). *2 Simple ways to code linked lists in Python* [Video]. YouTube. https://www.youtube.com/watch?v=6sBsF13n5ig