

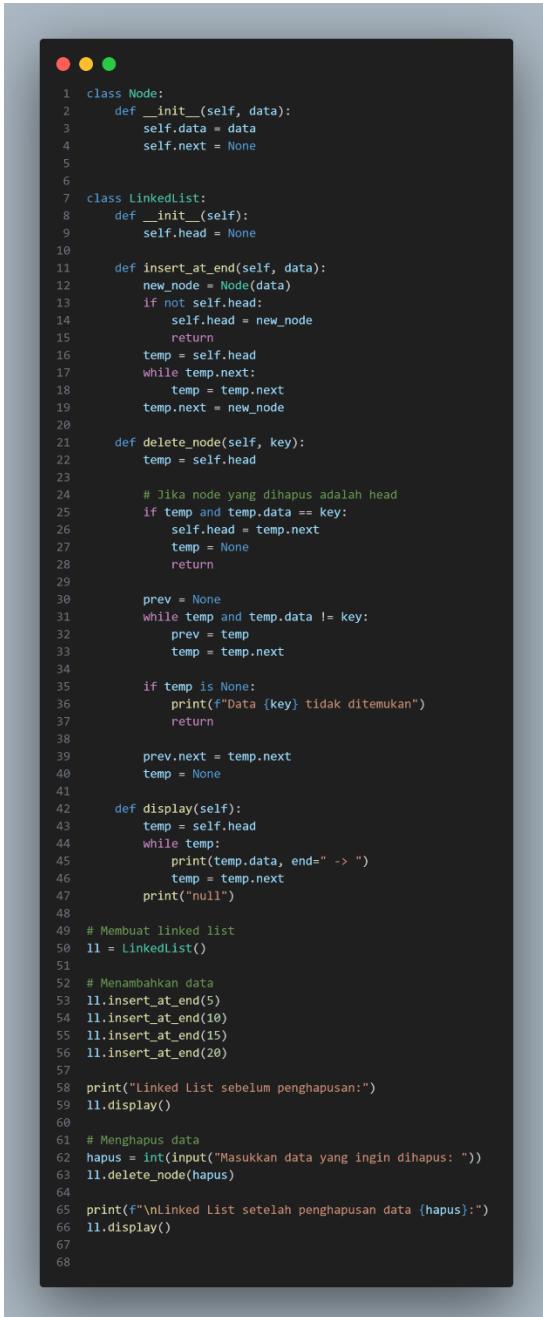
Pertemuan 3

Rafif Muhammad Faiz

J0403251024

TPL B/ P2

1. Tugas 1



```
1  class Node:
2      def __init__(self, data):
3          self.data = data
4          self.next = None
5
6
7  class LinkedList:
8      def __init__(self):
9          self.head = None
10
11     def insert_at_end(self, data):
12         new_node = Node(data)
13         if not self.head:
14             self.head = new_node
15             return
16         temp = self.head
17         while temp.next:
18             temp = temp.next
19         temp.next = new_node
20
21     def delete_node(self, key):
22         temp = self.head
23
24         # Jika node yang dihapus adalah head
25         if temp and temp.data == key:
26             self.head = temp.next
27             temp = None
28             return
29
30         prev = None
31         while temp and temp.data != key:
32             prev = temp
33             temp = temp.next
34
35         if temp is None:
36             print(f"Data {key} tidak ditemukan")
37             return
38
39         prev.next = temp.next
40         temp = None
41
42     def display(self):
43         temp = self.head
44         while temp:
45             print(temp.data, end=" -> ")
46             temp = temp.next
47         print("null")
48
49 # Membuat linked list
50 ll = LinkedList()
51
52 # Menambahkan data
53 ll.insert_at_end(5)
54 ll.insert_at_end(10)
55 ll.insert_at_end(15)
56 ll.insert_at_end(20)
57
58 print("Linked List sebelum penghapusan:")
59 ll.display()
60
61 # Menghapus data
62 hapus = int(input("Masukkan data yang ingin dihapus: "))
63 ll.delete_node(hapus)
64
65 print(f"\nLinked List setelah penghapusan data {hapus}:")
66 ll.display()
67
68
```

2. Tugas 2

```
1  class Node:
2      def __init__(self, data):
3          self.data = data
4          self.next = None
5
6
7  class CircularSinglyLinkedList:
8      def __init__(self):
9          self.head = None
10
11     def insert_at_end(self, data):
12         new_node = Node(data)
13         if not self.head:
14             self.head = new_node
15             new_node.next = self.head
16         else:
17             temp = self.head
18             while temp.next != self.head:
19                 temp = temp.next
20             temp.next = new_node
21             new_node.next = self.head
22
23     def search(self, key):
24         if not self.head:
25             print("Circular Linked List kosong. Tidak ada elemen yang bisa dicari.")
26             return
27
28         temp = self.head
29         while True:
30             if temp.data == key:
31                 print(f"Elemen {key} ditemukan dalam Circular Linked List.")
32                 return
33             temp = temp.next
34             if temp == self.head:
35                 break
36
37         print(f"Elemen {key} tidak ditemukan dalam Circular Linked List.")
38 cll = CircularSinglyLinkedList()
39
40 n = int(input("Masukkan jumlah elemen Circular Linked List: "))
41
42 for i in range(n):
43     data = int(input(f"Masukkan elemen ke-{i+1}: "))
44     cll.insert_at_end(data)
45
46 # Tetap jalan walaupun n = 0
47 cari = int(input("Masukkan elemen yang ingin dicari: "))
48 cll.search(cari)
49
```

3. Tugas 4



```
1  class Node:
2      def __init__(self, data):
3          self.data = data
4          self.next = None
5
6
7  class CircularSinglyLinkedList:
8      def __init__(self):
9          self.head = None
10
11     def insert_at_end(self, data):
12         # bikin node baru
13         new_node = Node(data)
14
15         # kalau list masih kosong
16         if not self.head:
17             self.head = new_node
18             new_node.next = self.head
19         else:
20             temp = self.head
21             while temp.next != self.head:
22                 temp = temp.next
23             temp.next = new_node
24             new_node.next = self.head
25
26     def display(self):
27         # kalau list kosong
28         if not self.head:
29             print("Circular Linked List kosong")
30             return
31
32         temp = self.head
33         while True:
34             print(temp.data, end=" -> ")
35             temp = temp.next
36             if temp == self.head:
37                 break
38         print("(kembali ke head)")
39
40     def merge(self, other):
41         # cek list kedua kosong atau tidak
42         if not self.head:
43             self.head = other.head
44             return
45
46         # cek list kedua kosong atau engga
47         if not other.head:
48             return
49
50         # mencari node terakhir list pertama
51         temp1 = self.head
52         while temp1.next != self.head:
53             temp1 = temp1.next
54
55         # mencari node terakhir list kedua
56         temp2 = other.head
57         while temp2.next != other.head:
58             temp2 = temp2.next
59
60         # menyambungkan kedua list
61         temp1.next = other.head
62         temp2.next = self.head
63
64 # Linked List pertama
65 cll1 = CircularSinglyLinkedList()
66 n1 = int(input("\nMasukkan jumlah elemen Circular Linked List 1: "))
67
68 for i in range(n1):
69     data = int(input(f"\nMasukkan elemen ke-{i+1} list 1: "))
70     cll1.insert_at_end(data)
71
72 print("\nCircular Linked List 1:")
73 cll1.display()
74
75 # Linked List kedua
76 cll2 = CircularSinglyLinkedList()
77 n2 = int(input("\nMasukkan jumlah elemen Circular Linked List 2: "))
78
79 for i in range(n2):
80     data = int(input(f"\nMasukkan elemen ke-{i+1} list 2: "))
81     cll2.insert_at_end(data)
82
83 print("\nCircular Linked List 2:")
84 cll2.display()
85
86
87 # Merge kedua list
88 cll1.merge(cll2)
89
90 print("\nHasil penggabungan Circular Linked List:")
91 cll1.display()
92
```