Lab 11.5 – Data Structures with AI

Student Name: SANJAY KARUPOTHULA
Roll No: 2303A52337
Program Name: B. Tech
Course Code: 23CS002PC304
Course Title: AI Assisted Coding
Year/Sem: III/II
Regulation: R23
Date and Day: Week 6 – Friday

Task 1 – Stack Implementation

```
class Stack:
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if not self.is_empty():
            return self.items.pop()
        return None

    def peek(self):
        if not self.is_empty():
            return self.items[-1]
        return None

    def is_empty(self):
        return len(self.items) == 0

s = Stack()
s.push(10)
s.push(20)
print(s.peek())
print(s.pop())
```

Screenshot – Stack Output:

main.py + 44e5mfgzp

```python
class Stack:
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if not self.is_empty():
            return self.items.pop()
        return None

    def peek(self):
        if not self.is_empty():
            return self.items[-1]
        return None

    def is_empty(self):
        return len(self.items) == 0


# Demo
s = Stack()
s.push(10)
s.push(20)

print("Top Element:", s.peek())
print("Popped:", s.pop())
print("Is Empty:", s.is_empty())
```

Output:

```
Top Element: 20
Popped: 20
Is Empty: False
```

Task 2 – Queue Implementation

```python
class Queue:
    def __init__(self):
        self.items = []

    def enqueue(self, item):
        self.items.append(item)

    def dequeue(self):
        if not self.is_empty():
            return self.items.pop(0)
        return None

    def peek(self):
        if not self.is_empty():
            return self.items[0]
        return None
```

```python
    def size(self):
        return len(self.items)

    def is_empty(self):
        return len(self.items) == 0

q = Queue()
q.enqueue(1)
q.enqueue(2)
print(q.dequeue())
```

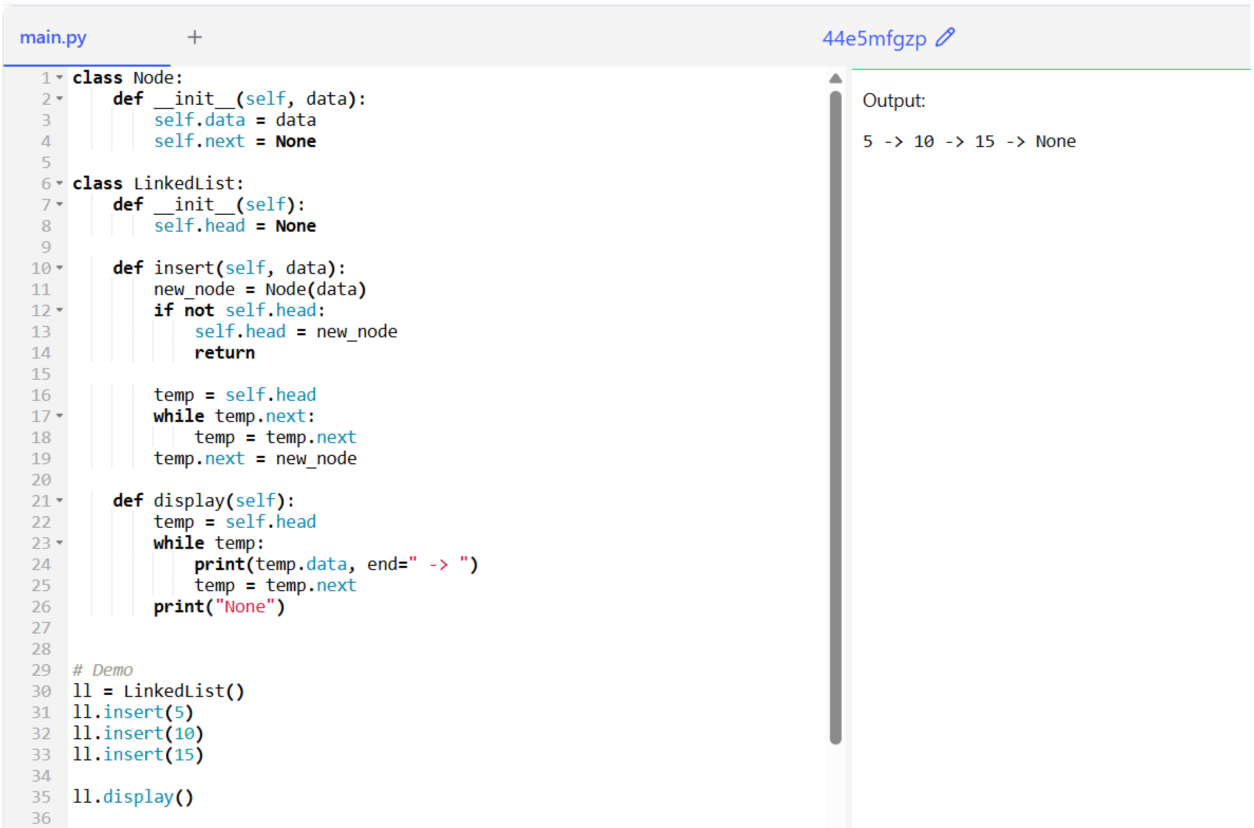
Screenshot – Queue Output:

Task 3 – Linked List

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

class LinkedList:
```

```python
    def __init__(self):
        self.head = None

    def insert(self, data):
        new_node = Node(data)
        if not self.head:
            self.head = new_node
            return
        temp = self.head
        while temp.next:
            temp = temp.next
        temp.next = new_node

    def display(self):
        temp = self.head
        while temp:
            print(temp.data, end=" -> ")
            temp = temp.next
        print("None")

ll = LinkedList()
ll.insert(5)
ll.insert(10)
ll.insert(15)
ll.display()
```

Screenshot – Linked List Output:

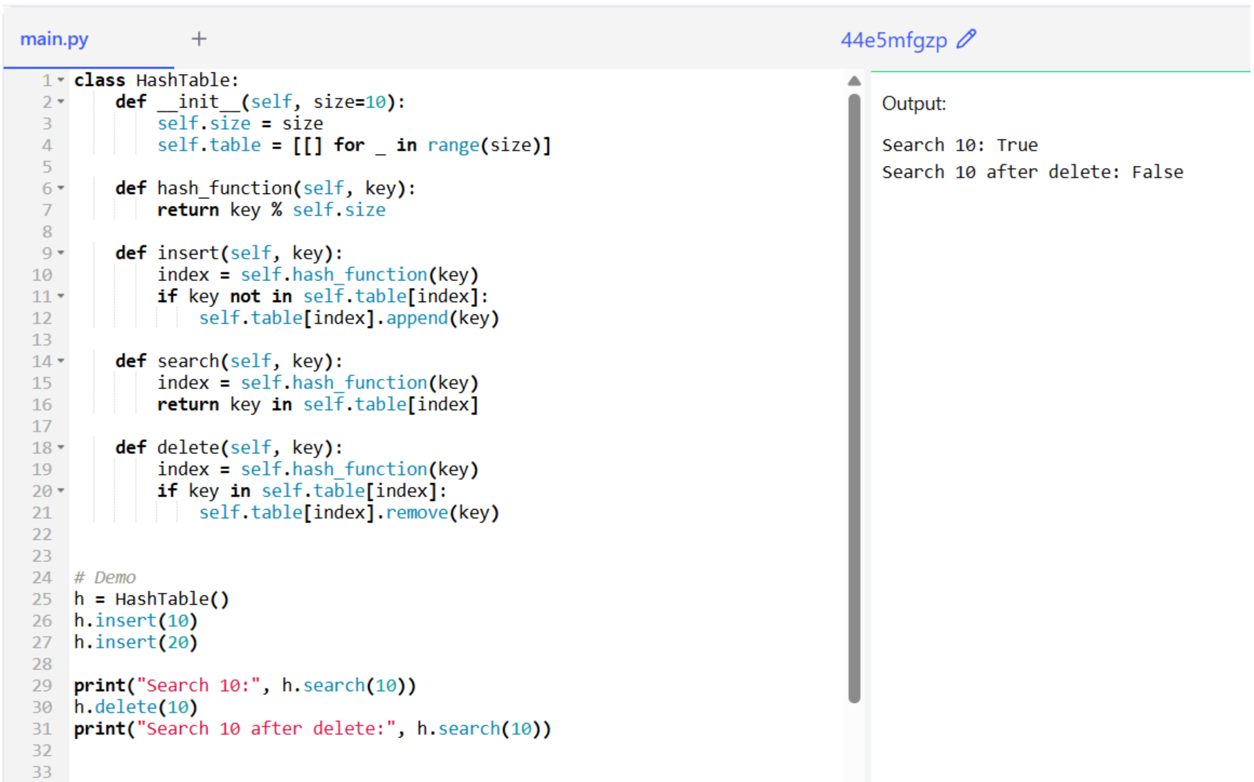


Task 4 – Hash Table

```python
class HashTable:
    def __init__(self, size=10):
        self.size = size
        self.table = [[] for _ in range(size)]

    def hash_function(self, key):
        return key % self.size

    def insert(self, key):
        index = self.hash_function(key)
        if key not in self.table[index]:
            self.table[index].append(key)

    def search(self, key):
        index = self.hash_function(key)
        return key in self.table[index]
```

```
    def delete(self, key):
        index = self.hash_function(key)
        if key in self.table[index]:
            self.table[index].remove(key)

h = HashTable()
h.insert(10)
h.insert(20)
print(h.search(10))
```

Screenshot – Hash Table Output:

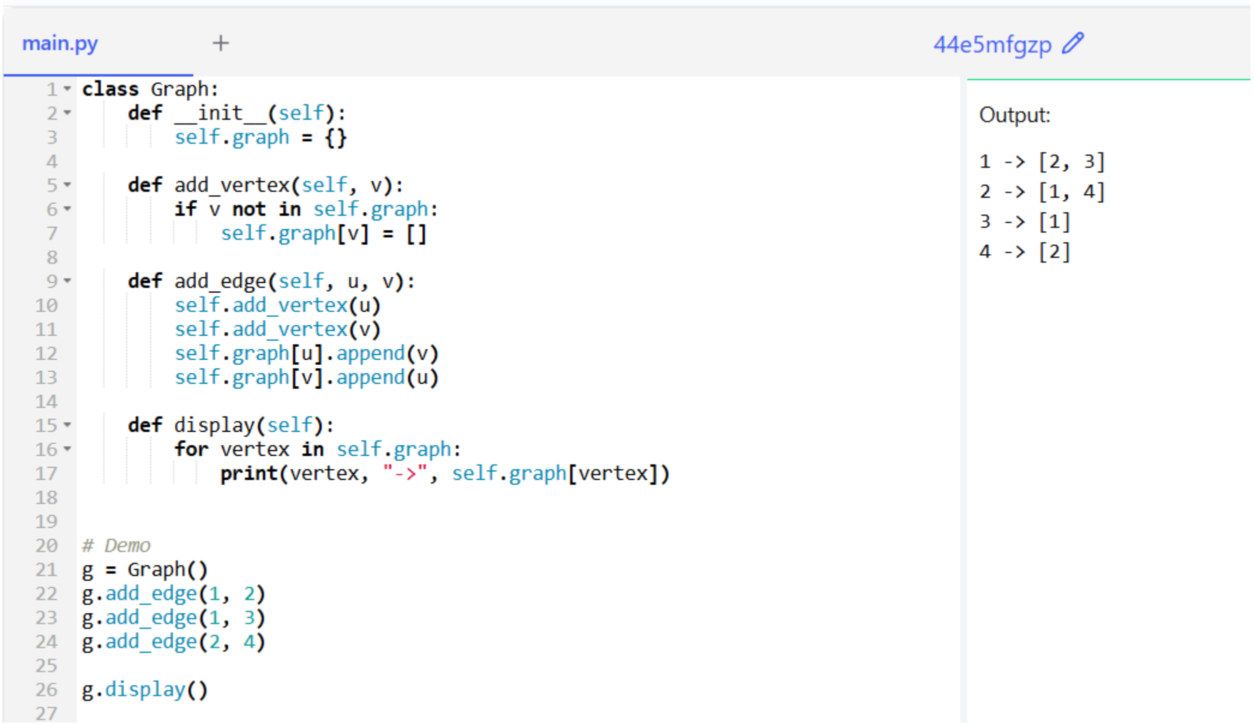


Task 5 – Graph

```
class Graph:
    def __init__(self):
        self.graph = {}

    def add_vertex(self, v):
        if v not in self.graph:
            self.graph[v] = []

    def add_edge(self, u, v):
        self.add_vertex(u)
        self.add_vertex(v)
```

```
            self.graph[u].append(v)
            self.graph[v].append(u)

        def display(self):
            for vertex in self.graph:
                print(vertex, "->", self.graph[vertex])

g = Graph()
g.add_edge(1, 2)
g.add_edge(1, 3)
g.display()
```

Screenshot – Graph Output :



Result

All fundamental data structures were successfully implemented.