Assignment 2

Problem Statement:

Implement all the functions of a dictionary (ADT) using hashing and handle collisions using chaining with / without replacement. Data: Set of (key, value) pairs, Keys are mapped to values, Keys must be comparable, Keys must be unique. Standard Operations: Insert(key, value), Find(key), Delete(key)

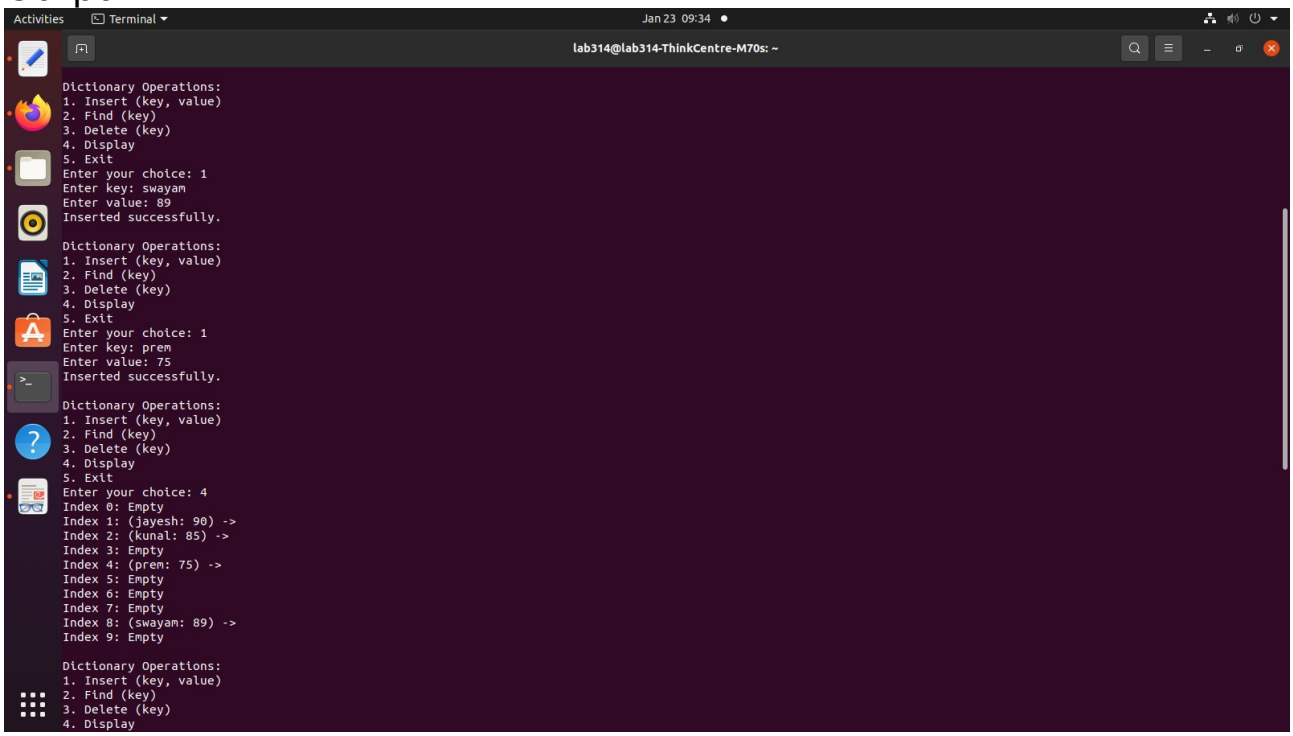
Source Code:

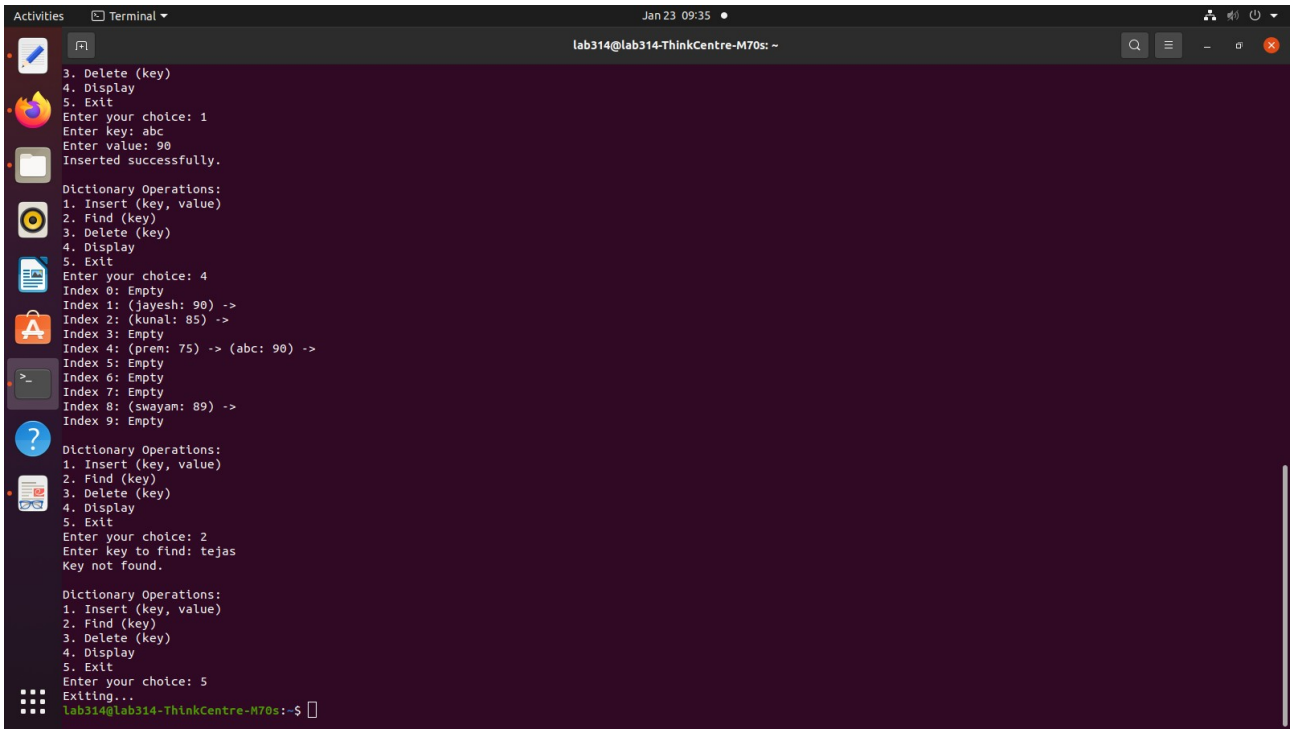
```
class Node:
    def __init__(self, key, value):
        self.key = key
        self.value = value
        self.next = None

class HashTable:
    def __init__(self, size=10):
        self.size = size
        self.table = [None] * self.size

    def _hash(self, key):
        # Simple hash function using the built-in Python hash and
modulo operator
        return hash(key) % self.size

    def insert(self, key, value):
        index = self._hash(key)
        new_node = Node(key, value)
        # If there is no chain at the index, add the new node directly
        if self.table[index] is None:
            self.table[index] = new_node
        else:
            # If collision occurs, handle using chaining (without
replacement)
            current = self.table[index]
            while current:
                if current.key == key:
                    # If key already exists, update value (with
replacement)
                    current.value = value
                    return
                if current.next is None:
```

```
                break
            current = current.next
        # If not found, append to the end of the chain
        current.next = new_node

    def find(self, key):
        index = self._hash(key)
        current = self.table[index]
        while current:
            if current.key == key:
                return current.value
            current = current.next
        return None  # Key not found

    def delete(self, key):
        index = self._hash(key)
        current = self.table[index]
        prev = None
        while current:
            if current.key == key:
                if prev is None:  # Deleting the first node in the chain
                    self.table[index] = current.next
                else:
                    prev.next = current.next
                return True
            prev = current
            current = current.next
        return False  # Key not found

    def display(self):
        for i in range(self.size):
            print(f"Index {i}:", end=" ")
            current = self.table[i]
            if current is None:
                print("Empty")
            else:
                while current:
                    print(f"({current.key}: {current.value})", end=" -> ")
                    current = current.next
                print()

# Main program to interact with the user
def main():
    hash_table = HashTable()
```

```
    while True:
        print("\nDictionary Operations:")
        print("1. Insert (key, value)")
        print("2. Find (key)")
        print("3. Delete (key)")
        print("4. Display")
        print("5. Exit")
        choice = int(input("Enter your choice: "))
        if choice == 1:
            key = input("Enter key: ")
            value = input("Enter value: ")
            hash_table.insert(key, value)
            print("Inserted successfully.")
        elif choice == 2:
            key = input("Enter key to find: ")
            result = hash_table.find(key)
            if result is None:
                print("Key not found.")
            else:
                print(f"Value for key {key}: {result}")
        elif choice == 3:
            key = input("Enter key to delete: ")
            if hash_table.delete(key):
                print(f"Key {key} deleted successfully.")
            else:
                print(f"Key {key} not found.")
        elif choice == 4:
            hash_table.display()
        elif choice == 5:
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

Output:


Activities
Terminal
Jan 23 09:34
lab314@lab314-ThinkCentre-M70s: ~
Dictionary Operations:
1. Insert (key, value)
2. Find (key)
3. Delete (key)
4. Display
5. Exit
Enter your choice: 1
Enter key: swayam
Enter value: 89
Inserted successfully.
Dictionary Operations:
1. Insert (key, value)
2. Find (key)
3. Delete (key)
4. Display
5. Exit
Enter your choice: 1
Enter key: prem
Enter value: 75
Inserted successfully.
Dictionary Operations:
1. Insert (key, value)
2. Find (key)
3. Delete (key)
4. Display
5. Exit
Enter your choice: 4
Index 0: Empty
Index 1: (jayesh: 90) ->
Index 2: (kunal: 85) ->
Index 3: Empty
Index 4: (prem: 75) ->
Index 5: Empty
Index 6: Empty
Index 7: Empty
Index 8: (swayam: 89) ->
Index 9: Empty
Dictionary Operations:
1. Insert (key, value)
2. Find (key)
3. Delete (key)
4. Display


Activities
Terminal
Jan 23 09:35
lab314@lab314-ThinkCentre-M70s: ~
3. Delete (key)
4. Display
5. Exit
Enter your choice: 1
Enter key: abc
Enter value: 90
Inserted successfully.
Dictionary Operations:
1. Insert (key, value)
2. Find (key)
3. Delete (key)
4. Display
5. Exit
Enter your choice: 4
Index 0: Empty
Index 1: (jayesh: 90) ->
Index 2: (kunal: 85) ->
Index 3: Empty
Index 4: (prem: 75) -> (abc: 90) ->
Index 5: Empty
Index 6: Empty
Index 7: Empty
Index 8: (swayam: 89) ->
Index 9: Empty
Dictionary Operations:
1. Insert (key, value)
2. Find (key)
3. Delete (key)
4. Display
5. Exit
Enter your choice: 2
Enter key to find: tejas
Key not found.
Dictionary Operations:
1. Insert (key, value)
2. Find (key)
3. Delete (key)
4. Display
5. Exit
Enter your choice: 5
Exiting...
lab314@lab314-ThinkCentre-M70s:~$