Assignment no.: 5

Name : Kapare Swapnali Namdev

Class : SE

Div : A

Roll no : A- 65

Problem statement :

Implementahashtableofsize10andusethedivisionmethodasahashfunction. Incaseof acollision,usechaining. Implementthefollowingoperations:

- Insert(key): Insertkey-valuepairsintothehashtable.
- Search(key):Searchforthevalueassociatedwithagivenkey.
- Delete(key):Deleteakey-valuepairfromthehashtable

Input :

```
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, value):
```

```
    index = self._hash_function(key)
    # Check if key already exists and update
    for pair in self.table[index]:
        if pair[0] == key:
            pair[1] = value
            print(f"Updated key {key} with value {value}")
            return
    # Otherwise, append new key-value pair
    self.table[index].append([key, value])
    print(f"Inserted key {key} with value {value}")

def search(self, key):
    index = self._hash_function(key)
    for pair in self.table[index]:
        if pair[0] == key:
            print(f"Found key {key} with value {pair[1]}")
            return pair[1]
    print(f"Key {key} not found")
    return None

def delete(self, key):
    index = self._hash_function(key)
    for i, pair in enumerate(self.table[index]):
        if pair[0] == key:
            del self.table[index][i]
```

```
            print(f"Deleted key {key}")
            return
        print(f"Key {key} not found for deletion")

    def display(self):
        print("Hash Table Contents:")
        for i, bucket in enumerate(self.table):
            print(f"Index {i}: {bucket}")

# 🔧 Demonstration
ht = HashTable()

# Insert operations
ht.insert(10, "Iris")
ht.insert(20, "Tulip")
ht.insert(30, "Lotus")
ht.insert(25, "Hibiscus")  # Will collide with key 5

# Search operations
ht.search(10)
ht.search(25)
ht.search(99)  # Not present

# Delete operations
ht.delete(20)
```

```
ht.delete(99)  # Not present

# Display final state
ht.display()
```

Output :

```
Inserted key 10 with value Iris
Inserted key 20 with value Tulip
Inserted key 30 with value Lotus
Inserted key 25 with value Hibiscus
Found key 10 with value Iris
Found key 25 with value Hibiscus
Key 99 not found
Deleted key 20
Key 99 not found for deletion
Hash Table Contents:
Index 0: [[10, 'Iris'], [30, 'Lotus']]
Index 1: []
Index 2: []
Index 3: []
Index 4: []
Index 5: [[25, 'Hibiscus']]
Index 6: []
```

Index 7: []

Index 8: []

Index 9: []