**Practical No.2**

**Title:**

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).

**Objective:**

* To implement all the functions of a dictionary (ADT) using hashing
* To handle collisions using chaining with / without replacement.

**Source Code:**

class HashTable:

def \_\_init\_\_(self, size=10):

self.size = size

self.table = [[] for \_ in range(size)]

def \_hash(self, key):

return hash(key) % self.size

def insert(self, key, value):

bucket = self.table[self.\_hash(key)]

if any(k == key for k, \_ in bucket):

raise ValueError(f"Key '{key}' already exists")

bucket.append((key, value))

def find(self, key):

bucket = self.table[self.\_hash(key)]

return next((v for k, v in bucket if k == key), None)

def delete(self, key):

bucket = self.table[self.\_hash(key)]

if not any(k == key for k, \_ in bucket):

raise KeyError(f"Key '{key}' not found")

bucket.remove(next((item for item in bucket if item[0] == key)))

def menu():

print("1. Insert\n2. Find\n3. Delete\n4. Exit")

hash\_table = HashTable()

while True:

menu()

choice = input("Enter your choice (1/2/3/4): ")

if choice == '4': break

if choice == '1':

try:

key, value = input("Enter key: "), input("Enter value: ")

hash\_table.insert(key, value)

print("Insertion successful!")

except ValueError as e:

print(f"ERROR! {e}")

elif choice == '2':

key = input("Enter key to find: ")

result = hash\_table.find(key)

print(f"Value for key '{key}': {result}") if result else print(f"Key '{key}' not found.")

elif choice == '3':

key = input("Enter key to delete: ")

try:

hash\_table.delete(key)

print(f"Key '{key}' deleted successfully.")

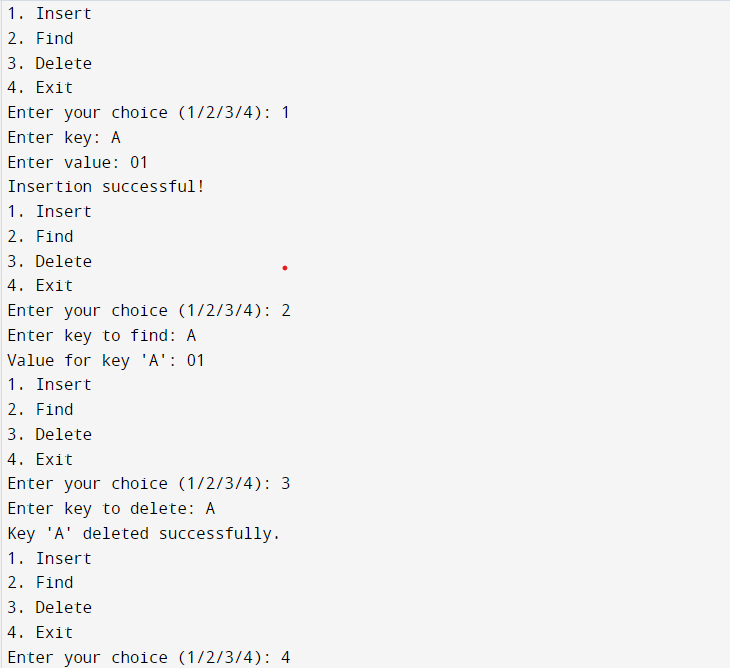
except KeyError as e:

print(f"ERROR! {e}")

else:

print("Invalid choice. Please enter a valid option.")

**Output:**

****