**Practical GroupA\_2**

**Aim:**

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)

**Code:**

class Data:

def \_\_init\_\_(self):

self.name = ""

self.name1 = ""

class Hashing:

def \_\_init\_\_(self):

self.n = 0

self.sum = 0

self.x = 0

self.c = 0

self.i = 0

self.j = 0

self.na = ""

self.na1 = ""

self.d = [Data() for \_ in range(10)]

def insert(self):

print("\nEnter no. of words :: ")

self.n = int(input())

for self.j in range(self.n):

print("\nEnter the word :: ")

self.na = input()

print("Enter the meaning :: ")

self.na1 = input()

self.sum = 0

for self.i in range(len(self.na)):

self.sum += ord(self.na[self.i])

self.x = (self.sum // len(self.na)) % 10

self.c = self.x

while True:

if self.d[self.x].name == "":

self.d[self.x].name = self.na

self.d[self.x].name1 = self.na1

break

self.x = (self.x + 1) % 10

if self.c == self.x:

print("\nHash table is full")

break

def search(self):

print("\nEnter word :: ")

self.na = input()

self.sum = 0

for self.i in range(len(self.na)):

self.sum += ord(self.na[self.i])

self.x = (self.sum // len(self.na)) % 10

self.c = self.x

while True:

if self.d[self.x].name == self.na:

print("\nKey :: ", self.d[self.x].name, "\nMeaning :: ", self.d[self.x].name1)

break

self.x = (self.x + 1) % 10

if self.c == self.x:

print("\nWord not found")

break

def delete(self):

print("\nEnter the word to be deleted :: ")

self.na = input()

self.sum = 0

for self.i in range(len(self.na)):

self.sum += ord(self.na[self.i])

self.x = (self.sum // len(self.na)) % 10

self.c = self.x

while True:

if self.d[self.x].name == self.na:

print("\nKeyword", self.d[self.x].name, "Deleted")

self.d[self.x].name = ""

self.d[self.x].name1 = ""

break

self.x = (self.x + 1) % 10

if self.c == self.x:

print("\nWord not found")

break

def display(self):

print("\nKey\tMeaning")

print("----------------")

for i in range(10):

if self.d[i].name != "":

print(self.d[i].name, "\t", self.d[i].name1)

print()

if \_\_name\_\_ == "\_\_main\_\_":

h = Hashing()

loop = 1

while loop == 1:

print("\n\n-------------------------")

print(" DICTIONARY ")

print("-------------------------")

print("1.Insert")

print("2.Display")

print("3.Search")

print("4.Delete")

print("5.Exit")

print("Enter :: ")

n = int(input())

if n == 1:

h.insert()

elif n == 2:

h.display()

elif n == 3:

h.search()

elif n == 4:

h.delete()

elif n == 5:

loop = 0

print("\n-------------------------")

else:

print("\nYou entered something wrong")