**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)**

**PROGRAM:**

table = []

b,totl = 0,0

bucket = {}

def create():

global b

b = int(input("Enter the table size : "))

for i in range(b):

table.append([None,-1])

bucket[i] = -1

def printtable():

global b

for i in range(b):

print("\t",table[i],end="\n")

print("")

def chaininsert(key):

global b,totl

hash = key%b

if (table[hash][0]==None):

table[hash][0] = key

bucket[key%b] = hash

else:

flag = 0

for i in range(0,b):

hash = (key+i)%b

if (table[hash][0]==None):

totl += 1

flag = 1

if bucket[key%b]!=1:

table[bucket[key%b]][1] = hash

bucket[key%b] = hash

table[hash][0] = key

break

if(flag==0):

print("Key : ",key," not inserted - table full .")

def chainsearch(key):

global b

hash = key%b

if (table[hash][0]==key):

print("Key : ",key," is found at index : ",hash)

else:

flag,i,chain = 0,0,table[hash][1]

while(table[hash][0]!=None and table[hash][0]%b != key%b):

hash = (key+i)%b

chain = table[hash][1]

if (table[hash][0]==key):

print("Key : ",key," is found at index : ",hash)

chain = -1

flag = 1

break

i += 1

while(chain!=-1):

if (table[chain][0]==key):

print("Key : ",key," is found at index : ",chain)

flag = 1

break

chain = table[chain][1]

if(flag==0):

print("Key : ",key," not found.")

def chaindelete(key):

global b

hash = key%b

if (table[hash][0]==key):

table[hash][0],table[hash][1] = None,-1

print("Key : ",key," was deleted from index : ",hash)

else:

flag,i,pchain,chain = 0,0,hash,table[hash][1]

while(table[hash][0]!=None and table[hash][0]%b != key%b):

hash = (key+i)%b

pchain = chain

chain = table[hash][1]

if (table[hash][0]==key):

table[pchain][1] = table[chain][1]

table[chain][0],table[chain][1]=None,-1

print("Key : ",key," was deleted from index : ",chain)

chain = -1

flag = 1

break

i += 1

while(chain!=-1):

if (table[chain][0]==key):

table[pchain][1] = table[chain][1]

table[chain][0],table[chain][1]=None,-1

print("Key : ",key," was deleted from index : ",chain)

flag = 1

break

pchain = chain

chain = table[chain][1]

if(flag==0):

print("Key : ",key," not found.")

create()

while(1):

ch = int(input("Enter \n\t1-Table \n\t0-EXIT : "))

if ch == 1 :

while(1):

ch2 = int(input("Enter \n1-Insert \n2-Search \n3-Delete \n4-Display\n\n0-BACK :"))

if ch2==1:

key = int(input("Enter the key to be inserted : "))

chaininsert(key)

printtable()

elif ch2==2:

key = int(input("Enter the key to be searched : "))

chainsearch(key)

printtable()

elif ch2==3:

key = int(input("Enter the key to be deleted : "))

chaindelete(key)

printtable()

elif ch2==4:

printtable()

elif ch2==0:

print("GOING BACK.")

printtable()

break

elif ch == 0:

print("EXITING")

printtable()

break

else:

printtable()