**Name: Affan Shaikh**

**Roll no: COB227**

**Code :**

class Hashtable:

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

self.m = int(input("Enter size of hash table: "))

self.hashTable = [None] \* self.m

self.elecount = 0

self.comparison = 0

print(self.hashTable)

defhashFunction(self, key):

return key % self.m

defisFull(self):

return self.elecount == self.m

deflinearProbe(self, key, data):

index = self.hashFunction(key)

compare = 0

while self.hashTable[index] is not None:

index = (index + 1) % self.m # Make sure index wraps around

compare += 1

self.hashTable[index] = [key, data]

self.elecount += 1

print("Data inserted at", index)

print(self.hashTable)

print("Number of comparisons:", compare)

defgetLinear(self, key, data):

index = self.hashFunction(key)

while self.hashTable[index] is not None:

if self.hashTable[index] == [key, data]:

return index

index = (index + 1) % self.m

return None

defquadraticProbe(self, key, data):

index = self.hashFunction(key)

compare = 0

i = 0

while self.hashTable[index] is not None:

index = (index + i \* i) % self.m

compare += 1

i += 1

self.hashTable[index] = [key, data]

self.elecount += 1

print("Data inserted at", index)

print(self.hashTable)

print("Number of comparisons:", compare)

defgetQuadratic(self, key, data):

index = self.hashFunction(key)

i = 0

while self.hashTable[index] is not None:

if self.hashTable[index] == [key, data]:

return index

i += 1

index = (index + i \* i) % self.m

return None

definsertViaLinear(self, key, data):

if self.isFull():

print("Table is full")

return False

index = self.hashFunction(key)

if self.hashTable[index] is None:

self.hashTable[index] = [key, data]

self.elecount += 1

print("Data inserted at", index)

print(self.hashTable)

else:

print("Collision occurred, applying Linear Probe method")

self.linearProbe(key, data)

definsertViaQuadratic(self, key, data):

if self.isFull():

print("Table is full")

return False

index = self.hashFunction(key)

if self.hashTable[index] is None:

self.hashTable[index] = [key, data]

self.elecount += 1

print("Data inserted at", index)

print(self.hashTable)

else:

print("Collision occurred, applying Quadratic Probe method")

self.quadraticProbe(key, data)

def menu():

obj = Hashtable()

ch = 0

while ch != 3:

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("1. Linear Probe ")

print("2. Quadratic Probe ")

print("3. Exit")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

ch = int(input("Enter Choice: "))

if ch == 1:

ch2 = 0

while ch2 != 3:

print("\*\* Insert \*\*")

print("\*\* Search \*\*")

print("\*\* Exit \*\*")

ch2 = int(input("Enter your choice: "))

if ch2 == 1:

a = int(input("Enter phone number: "))

b = str(input("Enter name: "))

obj.insertViaLinear(a, b)

elif ch2 == 2:

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

b = str(input("Enter name: "))

f = obj.getLinear(k, b)

if f is None:

print("Key not found")

else:

print("Key found at", f)

elifch == 2:

ch2 = 0

obj1 = Hashtable()

while ch2 != 3:

print("\*\* Insert \*\*")

print("\*\* Search \*\*")

print("\*\* Exit \*\*")

ch2 = int(input("Enter your choice: "))

if ch2 == 1:

a = int(input("Enter phone number: "))

b = str(input("Enter name: "))

obj1.insertViaQuadratic(a, b)

elif ch2 == 2:

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

b = str(input("Enter name: "))

f = obj1.getQuadratic(k, b)

if f is None:

print("Key not found")

else:

print("Key found at", f)

Hashtable.menu()

**Output :**

Enter size of hash table: 10

[None, None, None, None, None, None, None, None, None, None]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Linear Probe

2. Quadratic Probe

3. Exit

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter Choice: 1

\*\* Insert \*\*

\*\* Search \*\*

\*\* Exit \*\*

Enter your choice: 1

Enter phone number: 333

Enter name: d

Data inserted at 3

[None, None, None, [333, 'd'], None, None, None, None, None, None]

\*\* Insert \*\*

\*\* Search \*\*

\*\* Exit \*\*

Enter your choice: 1

Enter phone number: 333444

Enter name: t

Data inserted at 4

[None, None, None, [333, 'd'], [333444, 't'], None, None, None, None, None]

\*\* Insert \*\*

\*\* Search \*\*

\*\* Exit \*\*

Enter your choice: 2

Enter key to be searched: 333

Enter name: d

Key found at 3