Assignment No. 1

Name: Yogesh Giridhar Chimandare

Roll No: COA218

Programme:

```
class Hashtable:
    def __init__(self):
        self.m = int(input("Enter size of hash table: "))
        self.hashTable = [None] * self.m
        self.elecount = 0
        self.comparisons = 0
        print("Initial Table:", self.hashTable)

    def hashFunction(self, key):
        return key % self.m

    def isfull(self):
        return self.elecount == self.m

    def linearprobr(self, key, data):
        index = self.hashFunction(key)
        compare = 0
        start = index
        while self.hashTable[index] is not None:
            index = (index + 1) % self.m
            compare += 1
            if index == start:
                print("Hash table is full, cannot insert")
                return
```

```
    self.hashTable[index] = [key, data]
    self.elecount += 1
    print("Data inserted at", index)
    print(self.hashTable)
    print("No of comparisons =", compare)

def getlinear(self, key, data):
    index = self.hashFunction(key)
    start = index
    while self.hashTable[index] is not None:
        if self.hashTable[index] == [key, data]:
            return index
        index = (index + 1) % self.m
        if index == start:
            break
    return None

def quadraticprobr(self, key, data):
    index = self.hashFunction(key)
    compare = 0
    i = 1
    start = index
    while self.hashTable[index] is not None:
        index = (start + i * i) % self.m
        compare += 1
        i += 1
        if i == self.m:
            print("Hash table is full, cannot insert")
            return
    self.hashTable[index] = [key, data]
    self.elecount += 1
```

```
    print("Data inserted at", index)
    print(self.hashTable)
    print("No of comparisons =", compare)

def getQuadratic(self, key, data):
    index = self.hashFunction(key)
    i = 1
    start = index
    while self.hashTable[index] is not None:
        if self.hashTable[index] == [key, data]:
            return index
        index = (start + i * i) % self.m
        i += 1
        if i == self.m:
            break
    return None

def insertvialinear(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 Probing")
        self.linearprobr(key, data)
```

```
    def insertviaQuadratic(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 Probing")
            self.quadraticprobr(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("1. Insert")
                print("2. Search")
```

```
        print("3. Exit")
        ch2 = int(input("Enter your choice: "))
        if ch2 == 1:
            a = int(input("Enter key: "))
            b = input("Enter name: ")
            obj.insertvialinear(a, b)
        elif ch2 == 2:
            k = int(input("Enter key to be searched: "))
            b = input("Enter name: ")
            f = obj.getlinear(k, b)
            if f is None:
                print("Key not found")
            else:
                print("Key found at", f)

    elif ch == 2:
        obj1 = Hashtable()
        ch2 = 0
        while ch2 != 3:
            print("1. Insert")
            print("2. Search")
            print("3. Exit")
            ch2 = int(input("Enter your choice: "))
            if ch2 == 1:
                a = int(input("Enter key: "))
                b = input("Enter name: ")
                obj1.insertviaQuadratic(a, b)
            elif ch2 == 2:
                k = int(input("Enter key to be searched: "))
                b = input("Enter name: ")
                f = obj1.getQuadratic(k, b)
```

```
        if f is None:
            print("Key not found")
        else:
            print("Key found at", f)

menu()
```

Output: