Practical No. 1 ;Telephone

```
class Person:
    def __init__(self):
        self.name = ""
        self.telephone_number = 0

    def set_data(self):
        self.name = input("Enter name: ")
        self.telephone_number = int(input("Enter telephone number: "))

    def display_data(self):
        print(f"Name: {self.name}")
        print(f"Telephone number: {self.telephone_number}")


def hash_key(key, size):
    return key % size


class HashTableLinearProbing:
    def __init__(self, size=10):
        self.ht = [None] * size
        self.size = size
        self.comparison = [0] * size

    def insert_data(self, person):
        comp = 1
        index = hash_key(person.telephone_number, self.size)
        original_index = index

        while self.ht[index] is not None:
            index = (index + 1) % self.size
            comp += 1
            if index == original_index:
                print("Hash table is full.")
                return

        self.ht[index] = person
        self.comparison[index] = comp
        print(f"Name: {person.name} entered with telephone number:
{person.telephone_number} with comparisons: {comp}")
```

```
        def search_data(self, telephone):
            index = hash_key(telephone, self.size)
            original_index = index
            while self.ht[index] is not None:
                if self.ht[index].telephone_number == telephone:
                    print(f"Number {telephone} found with name {self.ht[index].name} at Index:
{index}")
                    return True
                index = (index + 1) % self.size
                if index == original_index:
                    break
            print(f"Number {telephone} not found in the hash table.")
            return False

        def delete_data(self, telephone):
            index = hash_key(telephone, self.size)
            original_index = index
            while self.ht[index] is not None:
                if self.ht[index].telephone_number == telephone:
                    self.ht[index] = None
                    return True
                index = (index + 1) % self.size
                if index == original_index:
                    break
            print(f"Number {telephone} not found in the hash table.")
            return False

        def display(self):
            print("Index\t| Name\t\t| Telephone Number\t| Comparisons")
            for i in range(self.size):
                if self.ht[i] is not None:
                    print(f"{i}\t| {self.ht[i].name}\t| {self.ht[i].telephone_number}\t\t|
{self.comparison[i]}")
                else:
                    print(f"{i}\t| Empty")


    class HashTableQuadraticProbing:
        def __init__(self, size=10):
            self.ht = [None] * size
            self.size = size
            self.comparison = [0] * size
```

```
        def insert_data(self, person):
            comp = 1
            index = hash_key(person.telephone_number, self.size)
            original_index = index
            i = 1

            while self.ht[index] is not None:
                index = (original_index + i * i) % self.size
                comp += 1
                i += 1
                if index == original_index:
                    print("Hash table is full.")
                    return

            self.ht[index] = person
            self.comparison[index] = comp
            print(f"Name: {person.name} entered with telephone number:
{person.telephone_number} with comparisons: {comp}")

        def search_data(self, telephone):
            index = hash_key(telephone, self.size)
            original_index = index
            i = 0

            while self.ht[index] is not None:
                if self.ht[index].telephone_number == telephone:
                    print(f"Number {telephone} found with name {self.ht[index].name} at Index:
{index}")
                    return True
                i += 1
                index = (original_index + i * i) % self.size
                if index == original_index:
                    break
            print(f"Number {telephone} not found in the hash table.")
            return False

        def delete_data(self, telephone):
            index = hash_key(telephone, self.size)
            original_index = index
            i = 0

            while self.ht[index] is not None:
                if self.ht[index].telephone_number == telephone:
                    self.ht[index] = None
```

```
                return True
            i += 1
            index = (original_index + i * i) % self.size
            if index == original_index:
                break
        print(f"Number {telephone} not found in the hash table.")
        return False

    def display(self):
        print("Index\t| Name\t\t| Telephone Number\t| Comparisons")
        for i in range(self.size):
            if self.ht[i] is not None:
                print(f"{i}\t| {self.ht[i].name}\t| {self.ht[i].telephone_number}\t\t|
{self.comparison[i]}")
            else:
                print(f"{i}\t| Empty")


def main():
    while True:
        print("----------------------------------------------------------")
        print("1. Linear Probing\n2. Quadratic Probing\n3. Exit")
        choice = int(input("Enter your choice: "))

        if choice == 1:
            h1 = HashTableLinearProbing()
            while True:
                print("----------------------------------------------------------")
                print("1. Insert\n2. Search\n3. Delete\n4. Display\n5. Exit")
                choice1 = int(input("Enter your choice: "))
                if choice1 == 1:
                    n = int(input("Enter number of persons: "))
                    for _ in range(n):
                        person = Person()
                        person.set_data()
                        h1.insert_data(person)
                elif choice1 == 2:
                    telenum = int(input("Enter telephone number to search: "))
                    h1.search_data(telenum)
                elif choice1 == 3:
                    telenum = int(input("Enter telephone number to delete: "))
                    if not h1.delete_data(telenum):
                        print("Not Found")
                elif choice1 == 4:
```

```
                h1.display()
            elif choice1 == 5:
                break
            else:
                print("Enter a valid choice.")

        elif choice == 2:
            h2 = HashTableQuadraticProbing()
            while True:
                print("----------------------------------------------------------")
                print("1. Insert\n2. Search\n3. Delete\n4. Display\n5. Exit")
                choice1 = int(input("Enter your choice: "))
                if choice1 == 1:
                    n = int(input("Enter number of persons: "))
                    for _ in range(n):
                        person = Person()
                        person.set_data()
                        h2.insert_data(person)
                elif choice1 == 2:
                    telenum = int(input("Enter telephone number to search: "))
                    h2.search_data(telenum)
                elif choice1 == 3:
                    telenum = int(input("Enter telephone number to delete: "))
                    if not h2.delete_data(telenum):
                        print("Not Found")
                elif choice1 == 4:
                    h2.display()
                elif choice1 == 5:
                    break
                else:
                    print("Enter a valid choice.")

        elif choice == 3:
            print("Exiting program.")
            break
        else:
            print("Please enter a valid choice.")


if __name__ == "__main__":
    main()
```