NAME: Pratiksha sudam nikam

Roll No: 03

PROBLEM STATEMENT:-Design and implement a hash table of fixed size. Use the division method for the hash function and resolve collisions using linear probing. Allow the user to perform the following operations:

- Insert a key
- Search for a key
- Delete a key
- Display the table

```
'''
class LinearProbingHashTable:
    def __init__(self, size=10):
        self.size = size
        self.table = [None] * size
        self.DELETED = "<DELETED>"

    def _hash_function(self, key):
        return key % self.size

    def insert(self, key):
        index = self._hash_function(key)
        original_index = index
        while self.table[index] not in (None, self.DELETED):
            if self.table[index] == key:
                print(f"Key {key} already exists at index {index}.")
                return
            index = (index + 1) % self.size
            if index == original_index:
                print("Hash table is full. Cannot insert.")
                return
        self.table[index] = key
        print(f"Inserted key {key} at index {index}.")

    def search(self, key):
        index = self._hash_function(key)
        original_index = index
        while self.table[index] is not None:
            if self.table[index] == key:
                print(f" Key {key} found at index {index}.")
                return index
            index = (index + 1) % self.size
            if index == original_index:
                break
        print(f" Key {key} not found.")
        return None

    def delete(self, key):
        index = self.search(key)
        if index is not None:
            self.table[index] = self.DELETED
            print(f" Key {key} deleted from index {index}.")

    def display(self):
        print("\nHash Table Status:")
        for i, key in enumerate(self.table):
            print(f"Index {i}: {key}")
        print("-" * 30)


ht = LinearProbingHashTable(size=10)
```

```
while True:
    print("\n===== MENU =====")
    print("1. Insert")
    print("2. Search")
    print("3. Delete")
    print("4. Display")
    print("5. Exit")
    choice = int(input("Enter your choice: "))

    if choice == 1:
        key = int(input("Enter key to insert: "))
        ht.insert(key)

    elif choice == 2:
            key = int(input("Enter key to search: "))
            ht.search(key)

    elif choice == 3:
            key = int(input("Enter key to delete: "))
            ht.delete(key)

    elif choice == 4:
        ht.display()

    elif choice == 5:
        print("...Exiting program...")
        break
    else:
        print("Invalid choice. Try again!")
```

```
'''

OUTPUT


gescoe@gescoe-OptiPlex-3020:~/Desktop/SE B $ python3 hash.py


===== MENU =====

1. Insert

2. Search

3. Delete

4. Display

5. Exit

Enter your choice: 1

Enter key to insert: 24

Inserted key 24 at index 4.


===== MENU =====

1. Insert

2. Search

3. Delete

4. Display

5. Exit

Enter your choice: 1

Enter key to insert: 34

Inserted key 34 at index 5.


===== MENU =====

1. Insert

2. Search

3. Delete
```

```
4. Display
5. Exit
Enter your choice: 4

Hash Table Status:
Index 0: None
Index 1: None
Index 2: None
Index 3: None
Index 4: 24
Index 5: 34
Index 6: None
Index 7: None
Index 8: None
Index 9: None
------------------------------

===== MENU =====
1. Insert
2. Search
3. Delete
4. Display
5. Exit
Enter your choice: 2
Enter key to search: 31
 Key 31 not found.

===== MENU =====
1. Insert
2. Search
```

```
3. Delete

4. Display

5. Exit

Enter your choice: 3

Enter key to delete: 24

 Key 24 found at index 4.

 Key 24 deleted from index 4.


===== MENU =====

1. Insert

2. Search

3. Delete

4. Display

5. Exit

Enter your choice: 4


Hash Table Status:

Index 0: None

Index 1: None

Index 2: None

Index 3: None

Index 4: <DELETED>

Index 5: 34

Index 6: None

Index 7: None

Index 8: None

Index 9: None

------------------------------


===== MENU =====
1. Insert

2. Search

3. Delete

4. Display

5. Exit

Enter your choice: 5

...Exiting program...
```