# Author : Shubham (Roll No.: 21118)

# DSA Assignment 04 : Hashing Techniques (Linear Probing without replacement)

class Record():

    def \_\_init\_\_(self, num, name):

        self.ph\_num, self.name = num, name

    def \_\_str\_\_(self):

        return "Phone Num: {}, Name: {}".format(self.ph\_num, self.name)

class HashTable():

    def \_\_init\_\_(self, D):

        self.MAX\_SIZE = D

        self.SIZE = 0

        self.table = [Record(-1, "DUMMY") for \_ in range(D)]

        self.is\_del = [0 for \_ in range(D)]

    def Print(self):

        i = 0

        for entry in self.table:

            print(i, entry)

            i += 1

    def getHashVal(self, key):

        return key % self.MAX\_SIZE

    # Search for key in HashTable, if found return record else return dummy record

    def search(self, key):

        if self.SIZE == 0:

            print ("Hashtable is empty.\n")

            return Record(-1, "DUMMY")

        idx = self.getHashVal(key)

        init = idx

        while True:

            if (self.table[idx].ph\_num == key):  # key found

                return self.table[idx]

            if (self.table[idx].ph\_num == -1 and not self.is\_del[idx]):  # empty slot

                return Record(-1, "DUMMY")

            idx = (idx+1) % self.MAX\_SIZE

            if (idx == init):  # if hashtable is full

                return Record(-1, "DUMMY")

    # Successful insertion returns True

    def insert(self, key, val):

        if self.SIZE == self.MAX\_SIZE:

            print("Hashtable is full.")

            return 0

        if (self.search(key).ph\_num != -1):

            print("Record is already Present.")

            return 0

        idx = self.getHashVal(key)

        # linear probing

        while self.table[idx].ph\_num != -1:

            idx = (idx+1) % self.MAX\_SIZE

        self.table[idx] = Record(key, val)

        self.SIZE += 1

        return 1

    # Successful deletion returns True

    def delete(self, key):

        if (self.search(key).ph\_num == -1):

            print("Record is not Present.")

            return 0

        idx = self.getHashVal(key)

        while True:

            if (self.table[idx].ph\_num == key):

                break

            idx = (idx+1) % self.MAX\_SIZE

        self.is\_del[idx] = 1

        self.table[idx] = Record(-1, "DUMMY")

        self.SIZE -= 1

        return 1

def main():

    ht = HashTable(10)

    while True:

        print('''Choose:

        \t1 for insertion

        \t2 for searching

        \t3 for deletion.

        \t0 to Exit.''')

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

        if not choice in [0, 1, 2, 3]:

            print("INVALID CHOICE.\n")

        elif choice == 0:

            break

        else:

            print("\n\nEnter Details")

            key = int(input("Enter Phone Number: "))

            if choice == 1:

                val = input("Enter Name: ")

            if choice == 1:

                if ht.insert(key, val):

                    print("INSERTION SUCCESSFUL.")

                else:

                    print("INSERTION FAILED.")

            elif choice == 2:

                record = ht.search(key)

                if (record.ph\_num == -1):

                    print("NOT FOUND.")

                else:

                    print ("FOUND. Details are:")

                    print(record)

            elif choice == 3:

                if ht.delete(key):

                    print("DELETION SUCCESSFUL.")

                else:

                    print("DELETION FAILED.")

        print()

main()