```
# Consider telephone book database of N clients. Make use of a hash table
implementation to quickly look up client's telephone number. Make use of two
collision handling techniques and compare them using number of comparisons
required to find a set of telephone numbers (use linear probing with replacement
and without replacement)

size = 10

class Person:
    isAtRightPosition = False
    def __init__(self, name="", no=0):
        self.name = name
        self.number = no


class TelephoneBook:
    directory = []
    currentSize = 0
    def __init__(self):
        for i in range(size):
            self.directory.append(Person())
    def hashFunction(self, no):
        k=0
        for i in range (1,5):
            k += (no%10)*i
            no = int(no/10)
        return (k%size)

    def printppl(self):
        for i  in range (size):
            print(i,self.directory[i].name, self.directory[i].number)


    def insertWithoutReplacement(self, personName,personNo):
        if(self.currentSize == size):
            print("No place availabe")
            return
        index = self.hashFunction(personNo)
        if(self.directory[index].number == 0):
            self.directory[index].number= personNo
            self.directory[index].name = personName
            print("Inserted")
            self.directory[index].isAtRightPosition = True
            self.currentSize +=1
            return
        else:
            while(self.directory[index].number != 0):
                index = (index+1)%10
                self.directory[index].number = personNo
                self.directory[index].name = personName
                print("Inserted")
                self.currentSize +=1
                return

    def search(self, key):
        index = self.hashFunction(key)
        if(self.directory[index].number == key):
            print("!!!!Found")
            print("name : {}  No: {}".format(self.directory[index].name,
self.directory[index].number))
            print("No of collisions : 0")
            return index
        count = 0
        while(self.directory[index].number != 0):
            if(self.directory[index].number == key):
                print("!!!!Found")
```

```
                print("name : {}  No: {}".format(self.directory[index].name,
self.directory[index].number))
                print("No of collisions : {}".find(count))
                return index
            count +=1
            index = (index+1)%10
        print("Key Not Present in the table ")
        return -1

    def deletePerson(self, key):
        index = self.search(key)
        if(index == -1):
            print("Key is not Present in the table ")
            return
        temp = index
        self.directory[index].name = ''
        self.directory[index].number=0
        self.directory[index].isAtRightPosition = False
        index +=1
        while(index != temp):
            if(self.hashFunction(self.directory[index].number) == temp):
                self.directory[temp].number = self.directory[index].number
                self.directory[temp].name = self.directory[index].name
                self.directory[temp].isAtRightPosition = True
                self.directory[index].isAtRightPosition = False
                self.directory[index].number = 0
                self.directory[index].name = ''
                print("Successfully Replaced ")
                self.currentSize -=1
                return
            index = (index+1)%10

    def insertWithReplacement(self,name, no):
        if(self.currentSize == size):
            print("No place for insertion ")
            return
        index = self.hashFunction(no)
        if(self.directory[index].number == 0):
            self.directory[index].number= no
            self.directory[index].name = name
            self.directory[index].isAtRightPosition = True
            print("Inserted")
            self.currentSize +=1
            return
        elif(self.directory[index].isAtRightPosition == False):
            temp = index
            index +=1
            while(index != temp and self.directory[index].isAtRightPosition ==
False):
                if(self.directory[index].number == 0):
                    self.directory[index].number = no
                    self.directory[index].name = name
                    print("Inserted")
                    self.currentSize   +=1
                    return
                index += (index +1)%size


a = TelephoneBook()

while(True):
    print("1.   Insert With Replacement")
    print("2.   Insert Without Replacement")
    print("3.   delete ")
    print("4.   Search")
```

```
print("5.   Print directory")
print("0.   Exit from program")
ch = int(input("Enter your Choice : "))

if(ch ==1):
    name = str(input("Enter the name of person - "))
    number = int(input("Enter the phone Number - "))
    a.insertWithReplacement(name,number)
elif(ch == 2):
    name = str(input("Enter the name of person - "))
    number = int(input("Enter the phone Number - "))
    a.insertWithoutReplacement(name, number)
elif(ch == 3):
    number = int(input("Enter the phone Number - "))
    a.deletePerson(number)
elif(ch == 4):
    number = int(input("Enter the phone Number - "))
    a.search(number)
elif(ch==5):
    print("Printing the directory ")
    a.printppl()
elif(ch == 0):
    print("Ending the program ")
    break
else :
    print("Wrong choice")
```