'''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'''

from Record import Record

from DoubleHash import doubleHashTable

def input\_record():

record=Record()

name=input('Enter Name ')

number=int(input('Enter Number '))

record.set\_name(name)

record.set\_number(number)

return record

class HashTable:

def \_\_init\_\_(self):

self.size=int(input('Enter size of contact book '))

self.elementcount=0

self.comparisons=0

self.table=list(None for i in range(self.size))

def isFull(self):

if (self.elementcount==self.size):

return True

else:

return False

def display(self):

print('\n')

for i in range(self.size):

print('Hash Value ',str(i)+'\t\t',str(self.table[i]))

print('\nThe total number of contacts are ',self.elementcount,'\n\n')

def hashfun(self,record):

return record % self.size

def insert(self,record):

if self.isFull()==True:

print('Table full')

return False

isStored=False

position=self.hashfun(record.get\_number())

if self.table[position]==None:

self.table[position]=record

print('Phone number of',record.get\_name(),'is at Hash Value ',str(position))

isStored=True

self.elementcount+=1

else:

print('Collision encountered for %s\'s number at position %s.\nFinding new position.'%(record.get\_name(),str(position)))

while self.table[position]!=None:

position+=1

if position>=self.size:

position=0

self.table[position]=record

print('Phone number of',record.get\_name(),'is at ',str(position))

isStored=True

self.elementcount+=1

return isStored

def search(self,record):

isFound=False

position=self.hashfun(record.get\_number())

self.comparisons+=1

if (self.table[position].get\_name()==record.get\_name() and self.table[position].get\_number()==record.get\_number()):

isFound==True

print(record,' found at position {}'.format(position),' total comparisons made are',str(1))

return position

else:

position+=1

if (position>=self.size-1):

position=0

while (self.table[position]!=None):

if (self.table[position].get\_name()==record.get\_name() and self.table[position].get\_number()==record.get\_number()):

isFound==True

i=self.comparisons+1

print(record,' found at position {}'.format(position),' total comparisons made are',str(i))

return position

position+=1

if position>=self.size-1:

position=0

self.comparisons+=1

if isFound==False:

print('Record not found')

return False

#Main

while(True):

ch=int(input('Please Enter Your Choice\n1.Linear Probing\t2.Double Hashing\n3.Exit '))

if (ch==1):

o1=HashTable()

while(True):

ch1=int(input('Please Enter Your Choice\n1. Add Contact\t\t2. Search Contact\n3. Display Contacts\t4. Go Back\n5. Exit '))

if (ch1==1):

record=input\_record()

o1.insert(record)

elif (ch1==2):

record=input\_record()

o1.search(record)

elif (ch1==3):

o1.display()

elif (ch1==4):

break

elif (ch1==5):

exit()

else:

print('Invalid Choice, Please Try Again')

elif (ch==2):

o2=doubleHashTable()

while(True):

ch2=int(input('Please Enter Your Choice\n1. Add Contact\t\t2. Search Contact\n3. Display Contacts\t4. Go Back\n5. Exit '))

if (ch2==1):

record=input\_record()

o2.insert(record)

elif (ch2==2):

record=input\_record()

o2.search(record)

elif (ch2==3):

o2.display()

elif (ch2==4):

break

elif (ch2==5):

exit()

else:

print('Invalid Choice, Please Try Again')

elif (ch==3):

exit()

else:

print('Invalid Choice, Please Try Again')