**Solution:**

def hash\_function(key, size):

sum = 0

for i in range(0, len(key)):

# print(key[i], ord(key[i]))

sum += ord(key[i])

print(sum)

if sum % 2 == 0:

return (sum/2) % size

else:

return sum % size

class Node:

def \_\_init\_\_(self, key, value):

self.key = key

self.value = value

self.next = None

class hash\_table:

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

self.size = size

self.table = [None] \* size

# use forward chaining

def insert(self, key, value):

index = hash\_function(key, self.size)

if self.table[index] == None:

self.table[index] = Node(key, value)

else:

temp = self.table[index]

while temp.next != None:

temp = temp.next

temp.next = Node(key, value)

def search(self, key):

index = hash\_function(key, self.size)

temp = self.table[index]

while temp != None:

if temp.key == key:

return temp.value

temp = temp.next

def delete(self, key):

index = hash\_function(key, self.size)

temp = self.table[index]

if temp.key == key:

self.table[index] = temp.next

return

while temp.next != None:

if temp.next.key == key:

temp.next = temp.next.next

return

temp = temp.next

Writing only the relevant functions should suffice for both sets, no need to write the whole class implementation.

**Marking rubric:**

1. Hash function-> 5
2. Insert/Delete-> total 7

Identifying the index-> 2

Insertion/deletion-> 3

Using forward chaining-> 2

1. Search-> 3