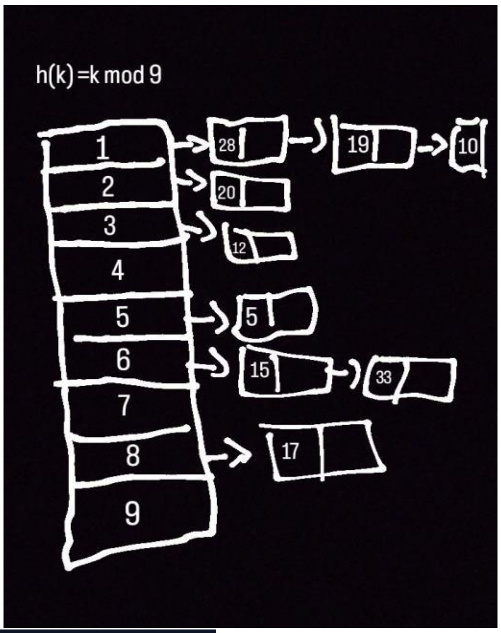
ALGO II
TP-TABLAS HASH

Exequiel Talfiti

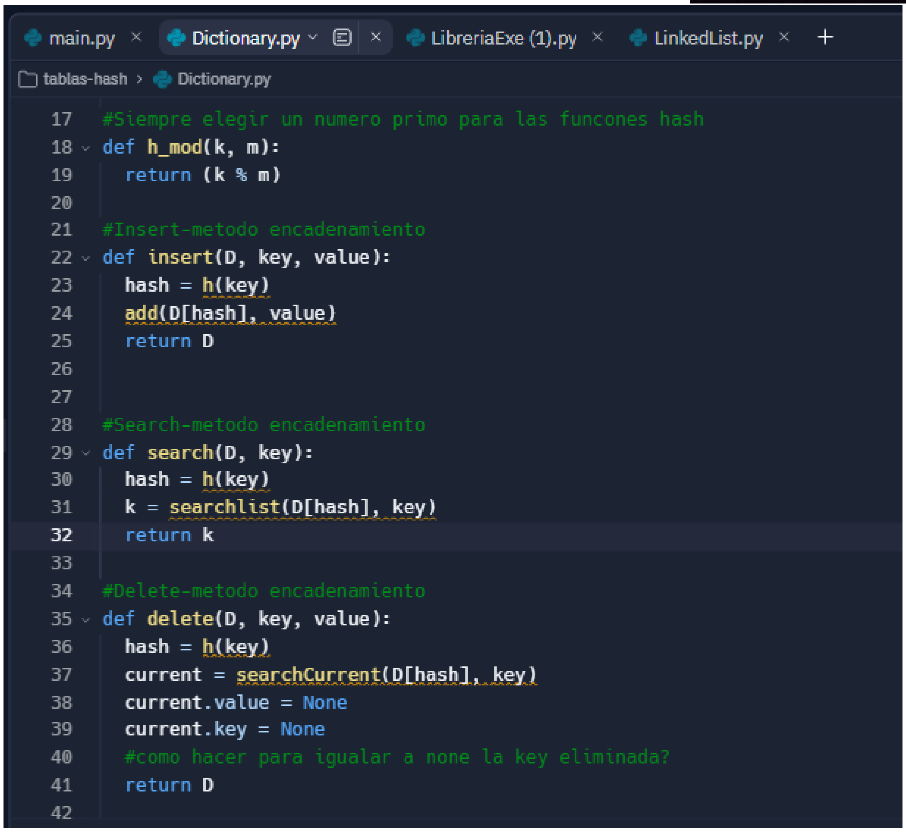
Parte 1

1)

h(5)=5 mod 9=5 list.add(D[h(5)],value)
h(28)=28 mod 9=1 list.add(D[h(28)],value)
h(19)=19 mod 9=1 list.add(D[h(19)],value)
h(15)=15 mod 9=6 list.add(D[h(15)],value)
h(20)=20 mod 9=2 list.add(D[h(20)],value)
h(33)=33 mod 9=6 list.add(D[h(33)],value)
h(12)=12 mod 9=3 list.add(D[h(12)],value)
h(17)=17 mod 9=8 list.add(D[h(17)],value)
h(10)=10 mod 9=1 list.add(D[h(10)],value)



2)



```python
#Siempre elegir un numero primo para las funcones hash
def h_mod(k, m):
  return (k % m)

#Insert-metodo encadenamiento
def insert(D, key, value):
  hash = h(key)
  add(D[hash], value)
  return D


#Search-metodo encadenamiento
def search(D, key):
  hash = h(key)
  k = searchlist(D[hash], key)
  return k

#Delete-metodo encadenamiento
def delete(D, key, value):
  hash = h(key)
  current = searchCurrent(D[hash], key)
  current.value = None
  current.key = None
  #como hacer para igualar a none la key eliminada?
  return D

```

Parte 2

3)

A=0,61 h(k)

=(m(kA mod 1))

h(61)=210

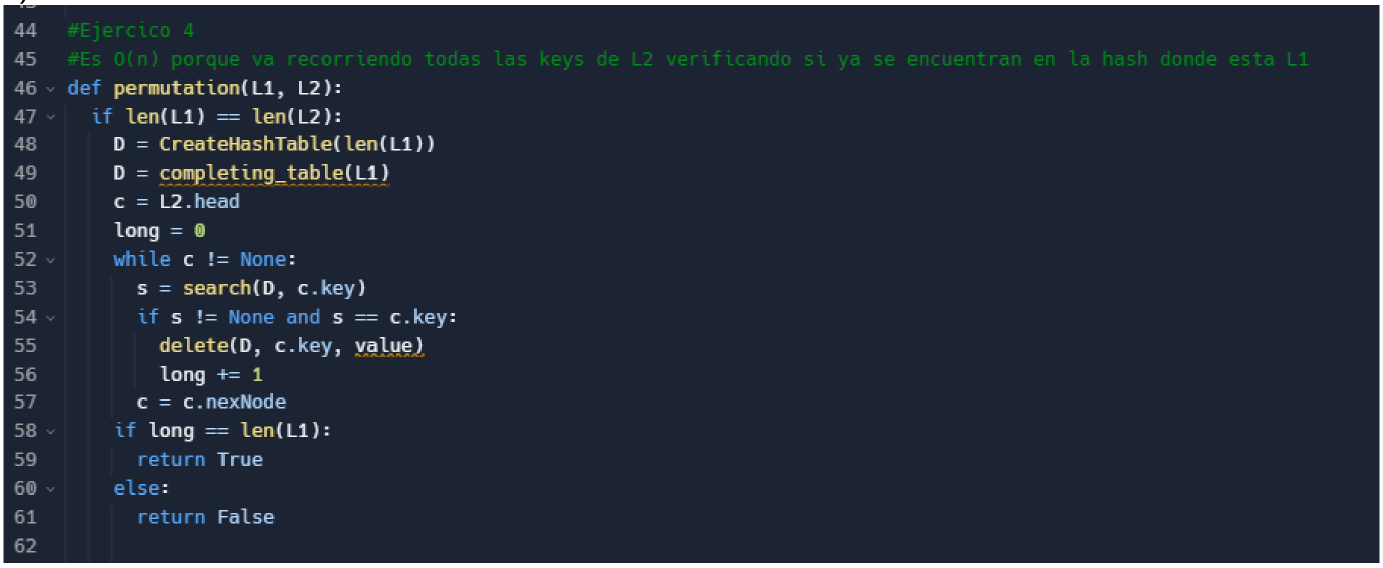
h(62)=820

h(63)=430

h(64)=40

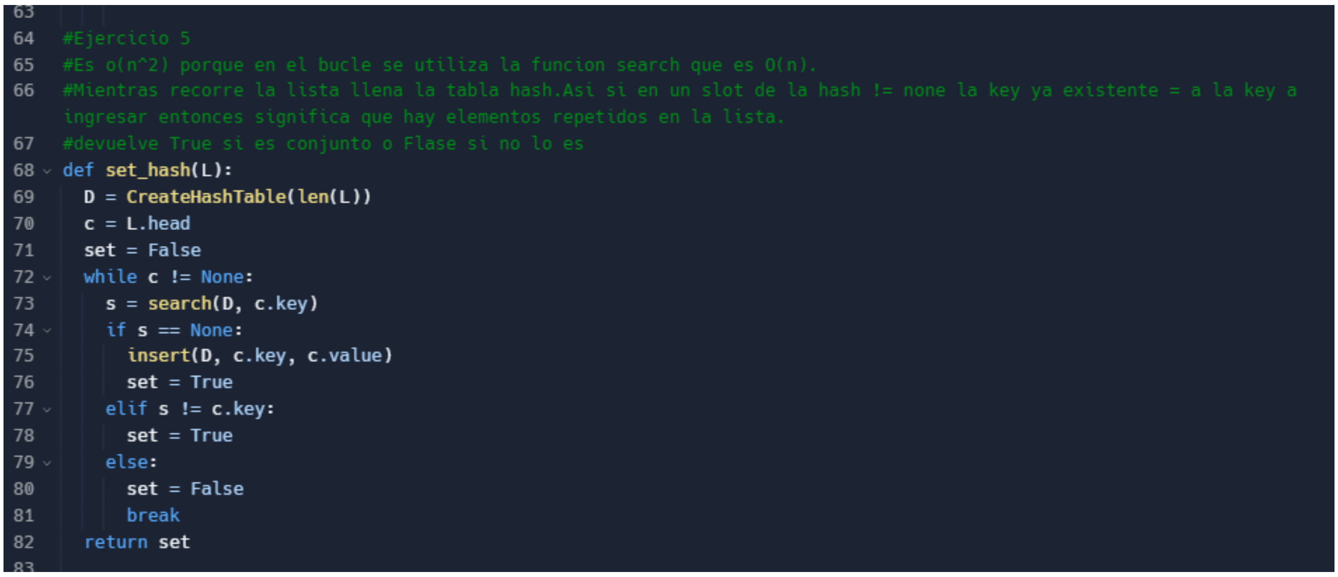
h(65)=650

4)



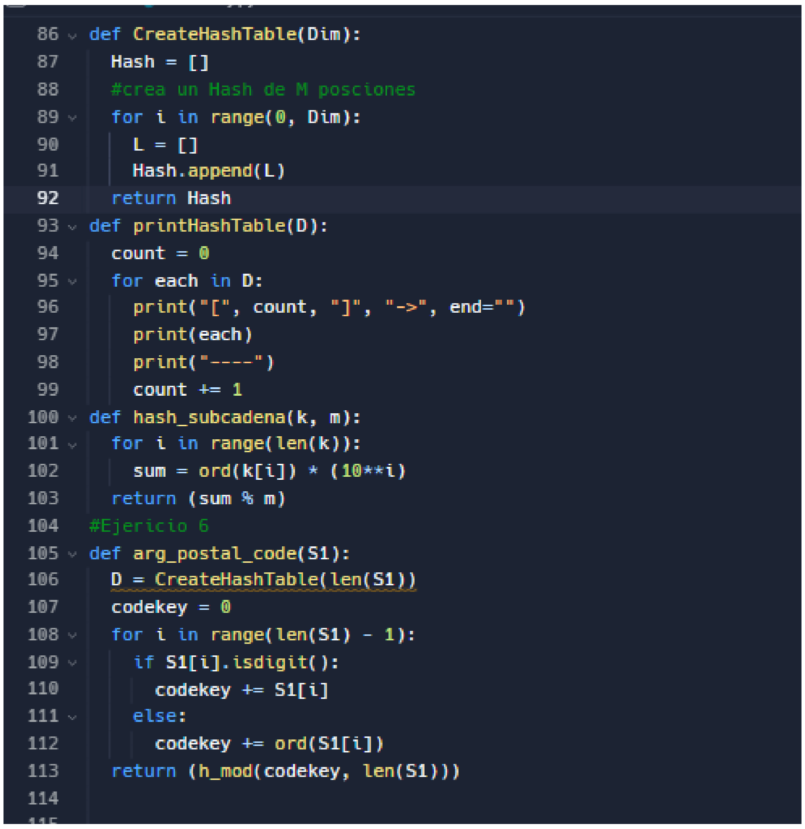
```python
#Ejercico 4
#Es O(n) porque va recorriendo todas las keys de L2 verificando si ya se encuentran en la hash donde esta L1
def permutation(L1, L2):
  if len(L1) == len(L2):
    D = CreateHashTable(len(L1))
    D = completing_table(L1)
    c = L2.head
    long = 0
    while c != None:
      s = search(D, c.key)
      if s != None and s == c.key:
        delete(D, c.key, value)
        long += 1
      c = c.nexNode
    if long == len(L1):
      return True
    else:
      return False
```

5)



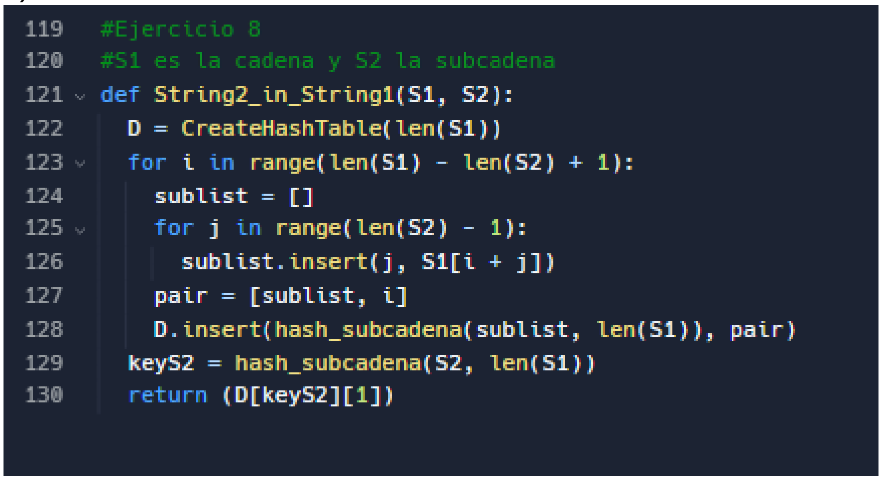
```python
#Ejercicio 5
#Es o(n^2) porque en el bucle se utiliza la funcion search que es O(n).
#Mientras recorre la lista llena la tabla hash.Asi si en un slot de la hash != none la key ya existente = a la key a ingresar entonces significa que hay elementos repetidos en la lista.
#devuelve True si es conjunto o Flase si no lo es
def set_hash(L):
  D = CreateHashTable(len(L))
  c = L.head
  set = False
  while c != None:
    s = search(D, c.key)
    if s == None:
      insert(D, c.key, c.value)
      set = True
    elif s != c.key:
      set = True
    else:
      set = False
      break
  return set
```

6)



```
def CreateHashTable(Dim):
  Hash = []
  #crea un Hash de M posciones
  for i in range(0, Dim):
    L = []
    Hash.append(L)
  return Hash
def printHashTable(D):
  count = 0
  for each in D:
    print("[", count, "]", "->", end="")
    print(each)
    print("----")
    count += 1
def hash_subcadena(k, m):
  for i in range(len(k)):
    sum = ord(k[i]) * (10**i)
  return (sum % m)
#Ejercicio 6
def arg_postal_code(S1):
  D = CreateHashTable(len(S1))
  codekey = 0
  for i in range(len(S1) - 1):
    if S1[i].isdigit():
      codekey += S1[i]
    else:
      codekey += ord(S1[i])
  return (h_mod(codekey, len(S1)))
```

8)



```
#Ejercicio 8
#S1 es la cadena y S2 la subcadena
def String2_in_String1(S1, S2):
  D = CreateHashTable(len(S1))
  for i in range(len(S1) - len(S2) + 1):
    sublist = []
    for j in range(len(S2) - 1):
      sublist.insert(j, S1[i + j])
    pair = [sublist, i]
    D.insert(hash_subcadena(sublist, len(S1)), pair)
  keyS2 = hash_subcadena(S2, len(S1))
  return (D[keyS2][1])
```