	Carátula para entrega de prácticas	Código	FODO-42
		Versión	01
		Página	1/1
		Sección ISO	
		Fecha de emisión	25 de junio de 2014
Secretaría/División: División de Ingeniería Eléctrica		Área/Departamento: Laboratorios de computación salas A y B	

Laboratorios de computación salas A y B

Profesor: Juan Carlos Catana Salazar

Asignatura: 1317: Estructura de datos y algoritmos II

Grupo: 8

No de Práctica(s): 6

Integrante(s) y Usuarios (HackerRank):

López Santibáñez Jiménez Luis Gerardo (HackerRank: LSJGerardo)

Semestre: 2017-1

Fecha de entrega: 16 de septiembre de 2016

Observaciones:

CALIFICACIÓN: _____

Objetivos

Implementación de búsqueda por transformación de llaves así como remover y agregar elementos de las mismas.

Código en la plataforma:

```
def agregar(cad):
    global ht
    ht[hashD(cad)].append(cad)  #Encontramos la casilla y
                                # lo agreamos como si fuera
                                #una lista ligada
def buscar(cad):
    global ht
    h = hashD(cad)
    for e in ht[h]:
        if(e == cad):   #Solo nos interesa saber si existe
            return h    # Y de ser así regresa la localidad
    else: return False

def borrar(cad):
    global ht
    index = buscar(cad)
    if (index):         #Si existe solo borramos la primer
        print "***"     # la primer coincidencia con .remove()
        print ht        # no tenemos que iterar
        print "\t borrando: " + cad + " en indice: " + str(index)
        ht[index].remove(cad)
        print ht
        return True
    else:
        print("No existe: " + cad)
        return False

def preHash(cad):   #Convertimos en ASCII las letras
    salida = ""
    for l in cad:
        salida+=str(ord(l))
    return int(salida)

```

```
34  def hashD(cad):     #El HASH real sucede aquí
35      global m
36      x = preHash(cad)    #Tomamos la cadena de numeros
37      A = 0.333           #Un numero aleatorio  y lo
38                          #regresamos de la siguiente manera
39      return int(( m*((x*A)%1) ))
40
41  m   = 13
42  ht  = [[] for j in range(m) ]
43  agregar("gerry")
44  agregar("perro")
45  agregar("123")
46  agregar("asd")
47  agregar("asde")
48  agregar("jose")
49  agregar("Jose")
50  print( borrar("123") )
```

Funcionamiento:

Utilizando una función de “Hash” asignamos una locación en nuestro arrego de tamaño “m” el cual se llenara de una manera algo uniforme y contendrá listas ligadas de los elementos que colisionen, aunque en Python se logra implementar de una manera mucho más sencilla por el uso de arreglos mutidimensionles que nos sirven como si fueran listas ligadas.

Conclusiones:

Es una forma muy interesante de utilizar “diccionarios” para agregar, buscar o borrar elementos de una “base de datos” de manera constante.