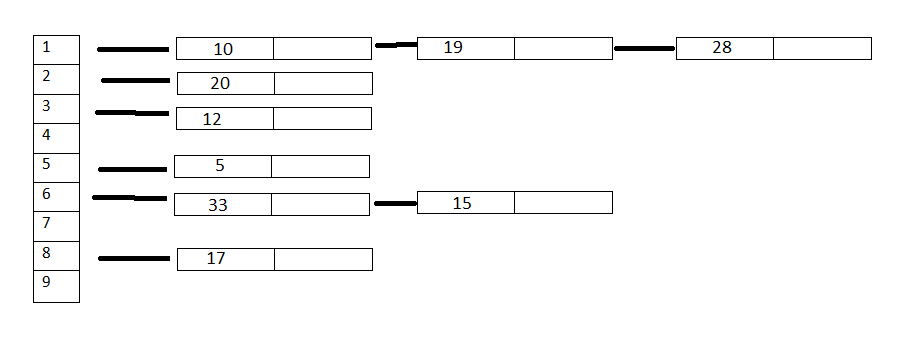
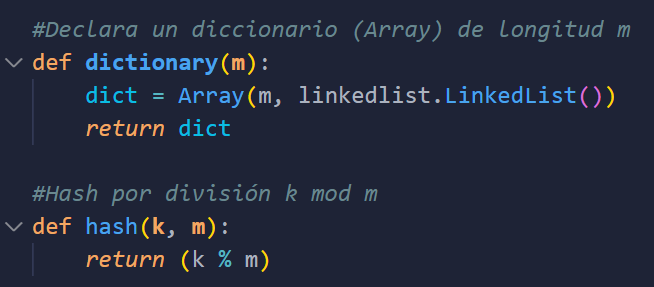
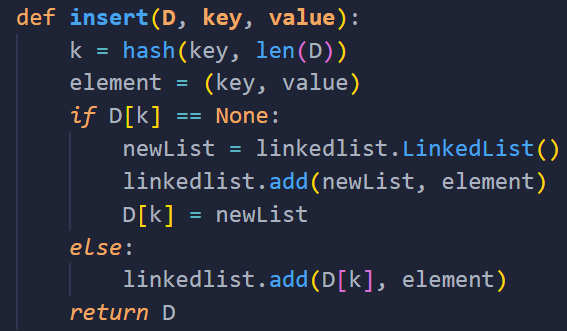
**Ejercicio 1)**



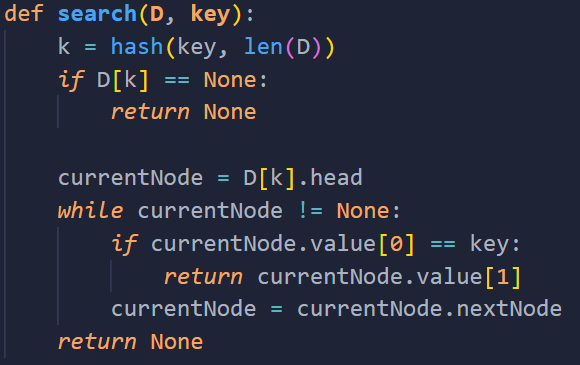
**Ejercicio 2)**



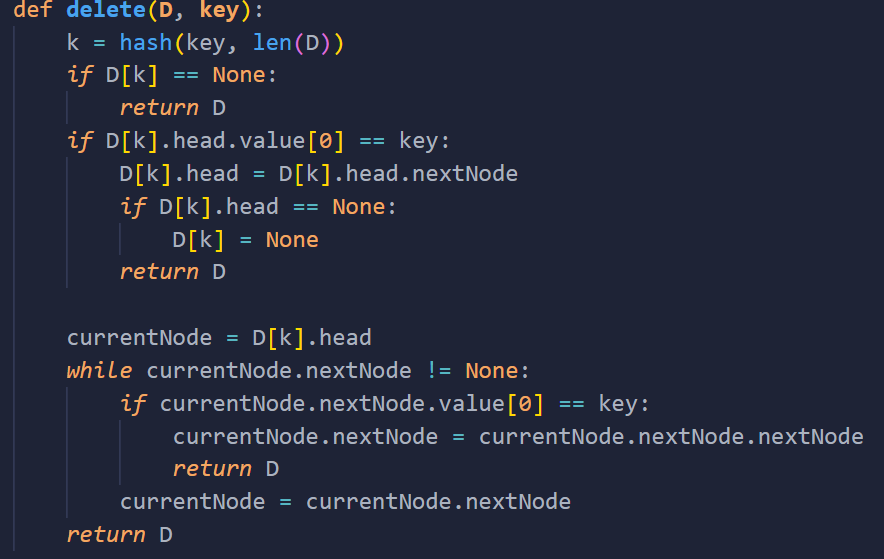
**Insert(D, key, value)**



**search(D, key)**



**delete(D, key)**



**Ejercicio 3)**

Función hash = m \* ((k \* (math.sqrt(5) -1) / 2) % 1)

Las claves luego de utilizar el hash son:

hash(61) = 700

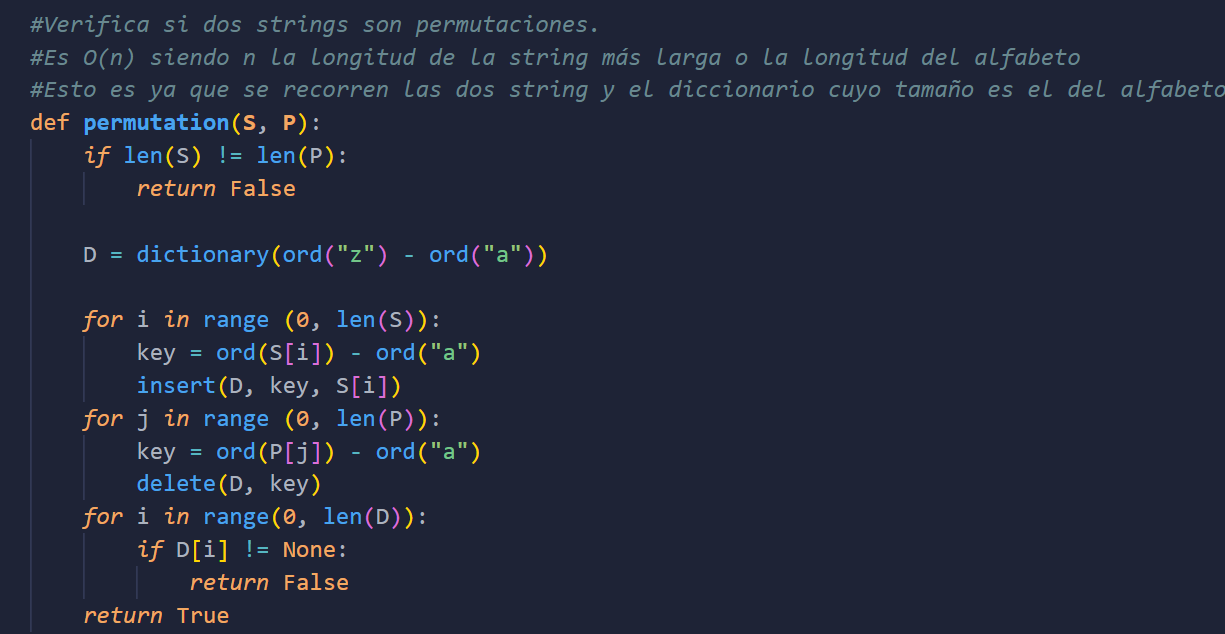
hash(62) = 318

hash(63) = 936

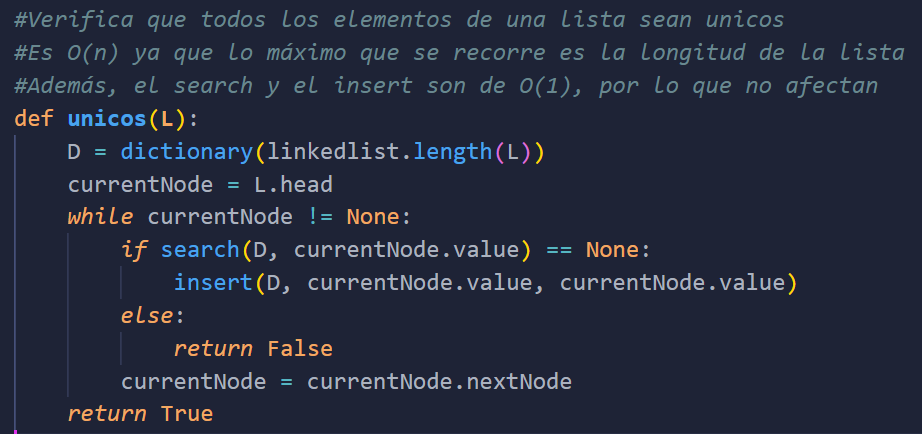
hash(64) = 554

hash(65) = 172

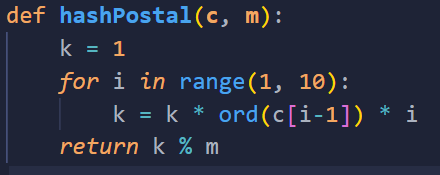
**Ejercicio 4)**



**Ejercicio 5)**

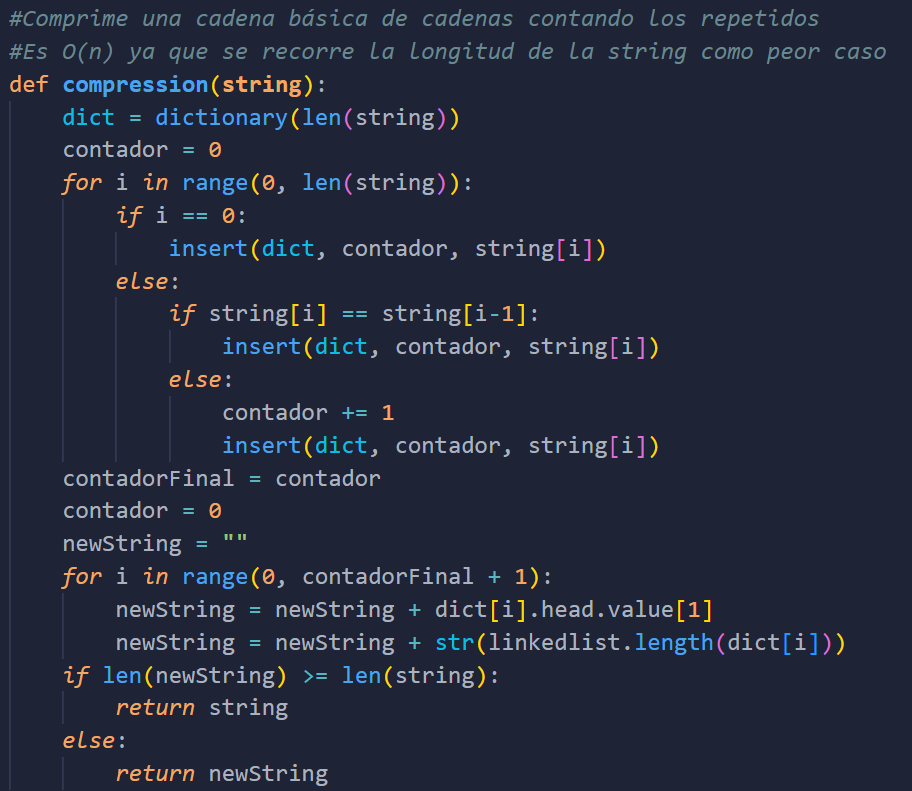


**Ejercicio 6)**

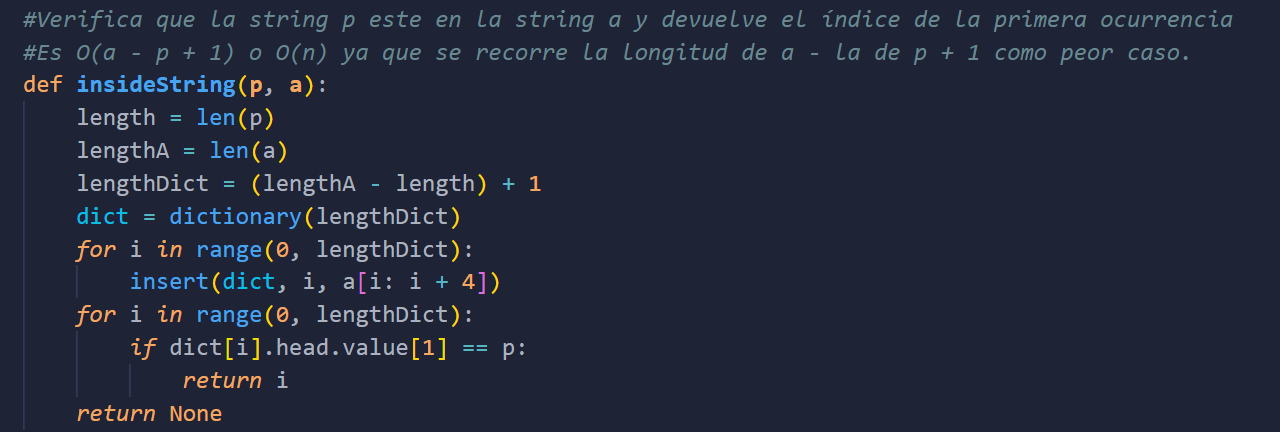


Recorre todo el código postal y por cada elemento de la string se toma su código ASCII y se lo multiplica por la posición en la que se encuentra. Finalmente, se reduce al tamaño de la tabla mediante k mod m.

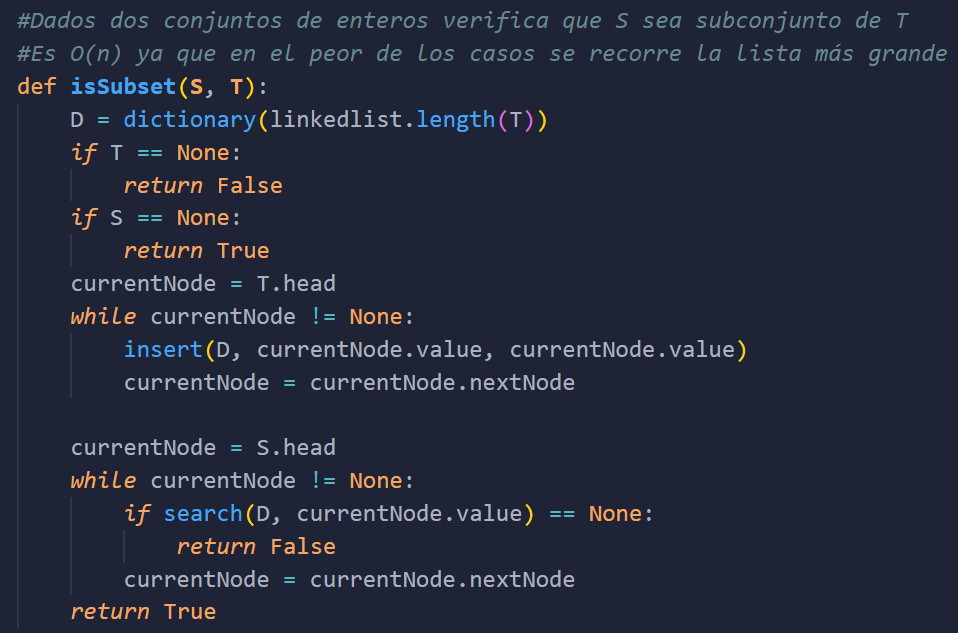
**Ejercicio 7)**



**Ejercicio 8)**

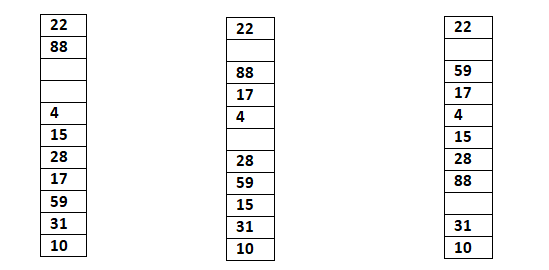


**Ejercicio 9)**



**Ejercicio 10)**

1. **2) 3)**



**Ejercicio 12)**

La correcta sería la tabla C. Las dos primeras se pueden descartar porque no tienen todos los elementos insertados, y, entre la D y la C podemos ver que es la C ya que utiliza la técnica pedida, es decir, calcula la posición mediante k % 10 y se fija si hay un elemento en ese lugar. Si existe, calcula una nueva posición mediante h’(x) = (h(x) + 1) % 10, hasta que encuentra una posición

**Ejercicio 13)**

La opción correcta es la C, ya que se utiliza la misma técnica que en el ejercicio anterior. Es decir, calcula la posición mediante k % 10 y se fija si hay un elemento en ese lugar. Si existe, calcula una nueva posición mediante h’(x) = (h(x) + 1) % 10, hasta que encuentra una posición. Si seguimos el patrón del C, podemos observar que queda como resultado la tabla brindada.