Práctica 3. Uso de Conjuntos y Diccionarios

2.1 Conjuntos

Ejercicio 1: Desarrollar un programa que solicite los nombres de pila de los 10 empleados de una empresa. A continuación, debe solicitar los nombres de 5 clientes de dicha empresa:

1. Mostrar los nombres de todas las personas sin repeticiones.
2. Informar qué nombres se repiten entre los empleados y clientes.
3. Informar de los nombres de empleados que no aparecen en los clientes

Ejercicio 2: Escribir una clase que permita la corrección ortográfica, que chequee el léxico de palabras a partir de un conjunto *W*. Debe implementar un método *check(s),* el cual realice el chequeo del *string s* con respecto al conjunto de palabras en *W*. Si *s* está en *W*, la palabra *s* es correcta lexicográficamente y devuelve una lista con una única palabra *[s]*. Si *s* no está en *W*, entonces debe devolver una lista con las palabras en *W* que pueden corregir *s*. Se deben contemplar palabras con dos caracteres adyacentes intercambiados, insertar un simple carácter entre dos adyacentes, borrado de un simple carácter, y sustitución de un carácter en una palabra. Considerar que las palabras son siempre en minúscula.

Ejercicio 3: Escribir un programa con el mismo propósito que el ejercicio anterior, pero ahora devolviendo todas aquellas palabras del conjunto W que son "similares" a la que se pasa como argumento al método *check*. Para ello, se propone hacer uso de la librería *difflib* de Python, que contiene una clase llamada *SequenceMatcher*, que facilita esta tarea. Investiga el uso de dicha clase para poder realizar el cálculo de la similitud entre dos palabras para implementar el ejercicio, y prueba su funcionamiento de forma adecuada, usando el conjunto W y un valor de similitud apropiados para obtener resultados significativos.

2.2 Diccionarios

Ejercicio 4: Escribir un programa que cree un diccionario de traducción español-inglés. El usuario introducirá las palabras en español e inglés separadas por dos puntos, y cada par <*palabra*>:<*traducción*> separados por comas. El programa debe crear un diccionario con las palabras y sus traducciones. Después pedirá una frase en español y utilizará el diccionario para traducirla palabra a palabra. Si una palabra no está en el diccionario debe dejarla sin traducir.

Ejercicio 5: Modifica la implementación del diccionario vista en clase, que usa una tabla de dispersión cerrada, para que use dispersión abierta. En cada "cubeta" se debe usar una lista para guardar los elementos. El número de cubetas será fijo y se pasará al constructor. Implementa el método mágico `__str__` para poder imprimir el diccionario completo. Prueba su funcionamiento insertando diversos elementos y probando todos los métodos incorporados.

Ejercicio 6: Escribir un programa que permita gestionar los alumnos de un colegio. Los alumnos se almacenarán en un diccionario en el que la clave de cada uno de ellos será su NIF, y el valor será una instancia de una clase que representa los datos del alumno (*nombre*, *dirección*, *teléfono*, *curso*, *repetidor*), donde *repetidor* tendrá el valor *True* si se trata de un alumno que está repitiendo curso. El programa debe preguntar al usuario por una opción del siguiente menú: *(1) Añadir alumno, (2) Eliminar alumno, (3) Mostrar alumno, (4) Listado de*

todos los alumnos, (5) Listado de alumnos repetidores, (6) Listado de alumnos por curso y *(7) Terminar*. Según la opción elegida el programa tendrá que hacer lo siguiente:

1. Preguntar los datos del alumno, crear un diccionario con los datos y añadirlo a la "base de datos".
2. Eliminar los datos de un alumno de la base de datos a partir de su NIF.
3. Mostrar los datos de un alumno a partir de su NIF.
4. Listar todos los alumnos mostrando su NIF y el resto de sus datos.
5. Mostrar la lista de alumnos repetidores mostrando sus datos.
6. Mostrar la lista de alumnos por curso mostrando sus datos.
7. Terminar el programa.