Práctica 4

1. Lea la implementación provista de hash tables[1], y el ejemplo presentado en el archivo `main.c`. Explique que pasa si se ejecuta `hashtable_insert` dos veces con la misma clave. Explique la razón por la cual se produce un error en la ejecución de `main.c`.

2. La implementación actual sufre de un problema: `hashtable_lookup` compara las claves usando la igualdad. Explique que problemas puede traer esto.
Modifique la implementación, agregando un miembro a la estructura `Hashtable` del tipo:

```
typedef int (*EqualsFunc)(void *, void *)
```

El objetivo de este miembro es determinar cuándo punteros a claves representan la misma clave. Modifique las funciones que sean necesarias para funcionar con este nuevo miembro.
Luego de haberlo implementado, compruebe que `main2.c` funciona correctamente.

3. Implementar las funciones `hashnat` y `hashstring` vistas en clase.

4. En lugar de utilizar funciones hash de la forma

```
typedef unsigned int (*HashFunc)(void *)
```

podemos cambiar la signatura a la forma

```
typedef unsigned int (*HashFunc)(void *, unsigned int)
```

donde el nuevo argumento es la cantidad de cubetas sobre la cuál se computa el hash. Actualice las funciones implementadas en el ejercicio anterior, y la implementación dada para funcionar con esta versión de `HashFunc`.

5. Modifique la implementación dada para soportar colisiones, utilizando la resolución de colisiones por encadenado.

6. Cuando una tabla hash tiene demasiados valores insertados es conveniente aumentar la cantidad de cubetas para disminuir la cantidad de colisiones. Para esto, es necesario recalcular el hash de todas las claves de la tabla para tomar en cuenta el nuevo máximo número de cubetas. Esta operación se denomina comúnmente *rehash*. Implemente la función `void hashtable_resize(Hashtable *)` que duplique la cantidad de slots de la tabla y reposicione todos los elementos en la tabla de acuerdo a la nueva posición que le asigne la función de hash.

7. Cree un programa que simule el uso de una agenda telefónica. Dicha agenda debe implementarse con una tabla hash donde las claves serán los nombres de las personas en la agenda y los valores serán los números telefónicos. Los comandos que deberá soportar por entrada estándar son:

Comando	Argumentos	Resultado	Ejemplo
insert	nombre, número	Inserta el par (nombre, número) indexado por nombre, modifica el número si ya existe el nombre en la agenda	insert Juan Perez, 4101010
delete	nombre	Borra la entrada perteneciente a nombre (si existe)	delete Juan Perez
search	nombre	Imprime el número de 'nombre' si está en la agenda	search Juan Perez

[1] `http://dcc.fceia.unr.edu.ar/~erivas/estructuras/src/hashtables/`