OBSERVACIONES DEL LA PRACTICA

Juan David Salguero, 201923136, J.salguero@uniandes.edu.co
David Molina, 202125176, d.molinad@uniandes.edu.co

```
    """
    Este indice crea un map cuya llave es el autor del libro
    """
    catalog['authors'] = mp.newMap(800,
                                   maptype='CHAINING',
                                   loadfactor=4.0,
                                   comparefunction=compareAuthorsByName)
```

1) ¿Qué estructura de datos se usa para este índice?

Se usa un mapa con un sistema de separate chaining. Esto significa que en el caso que dos valores con una misma llave se colisionen, estos valores se guardan bajo esta misma llave en un espacio llamado 'bucket'.

2) ¿Cuántos elementos se espera almacenar inicialmente?

800 elementos., es el valor correspondiente al primer parámetro de la función mp.newMap().

3) ¿Cuál es el tamaño de las tablas de hash para ***'years'*** y ***'authors'***?

Para 'years' seria el numero primo mas cercano a el tamaño de sus datos, 40, dos veces: **83**
De igual manera, para 'authors' seria el numero primo mas cercano a el tamaño de sus datos dos veces: **1601**

4) ¿Cuál es el factor de carga máximo?

Para 'years': **(0.5)**, lo que significa que un valor ocupa dos espacios en la lista.
Para 'authors': (**4**), lo que significa que un espacio en la lista puede tener hasta 4 valores.

```python
def addBook(catalog, book):
    """
    Esta funcion adiciona un libro a la lista de libros,
    adicionalmente lo guarda en un Map usando como llave su Id.
    Adicionalmente se guarda en el indice de autores, una referencia
    al libro.
    Finalmente crea una entrada en el Map de años, para indicar que este
    libro fue publicado en ese año.
    """
    lt.addLast(catalog['books'], book)
    mp.put(catalog['bookIds'], book['goodreads_book_id'], book)
    authors = book['authors'].split(",")  # Se obtienen los autores
    for author in authors:
        addBookAuthor(catalog, author.strip(), book)
    addBookYear(catalog, book)
```

5) ¿Qué hace la instrucción "**mp.put(...)**"?

En palabras de la librería: "Ingresa una pareja llave, valor a la tabla de hash. Si la llave ya existe en la tabla, se reemplaza el valor"
El primer argumento es el mapa donde se guarda la pareja (llave, valor). El segundo argumento corresponde a la llave descrita y el tercer argumento a su valor correspondiente.

6) ¿Qué papel cumple **"book['goodreads_book_id']"** en esa instrucción?

book['goodreads_book_id'] es el segundo argumento de la función, y por ende corresponde a la llave usada en la pareja que se guardará en el mapa.

7) ¿Qué papel cumple el tercer parámetro ***"book"*** en esa instrucción?

book es el tercer argumento de la función, y por ende corresponde al valor usado en la pareja que se guardará en el mapa.

```python
def getBooksByYear(catalog, year):
    """
    Retorna los libros publicados en un año
    """
    year = mp.get(catalog['years'], year)
    if year:
        return me.getValue(year)['books']
    return None
```

8) ¿Qué hace la instrucción **"mp.get(…)"**?

Busca en un mapa especificado (el primer argumento), una llave dada a la función (segundo argumento). Si la encuentra, retorna la pareja llave/valor correspondiente a la llave dada en el input..

9) ¿Qué papel cumple ***"year"*** en esa instrucción?

El objeto “year” funciona como una llave buscada. mp.get() itera catalog[“years“] en busca de esta llave.

10) ¿Qué hace la instrucción **“me.getValue(…)”**?

Dada una pareja llave/valor, pareja la cual es obtenida en este ejemplo por mp.get(), me.getValue() retorna el valor correspondiente a esta pareja.