Data Structures practice 3 report

Structs:

We followed a similar implementation for the index and list of deleted records:

```
typedef struct Book
{
    size_t offset;
    char *code;
    char *isbn;
    char *title;
    char *printedBy;
} Book;

/**index**/
typedef struct {
    int key; /** book isbn */
    long int offset; /** book isstoredindi skatthispostion */
    size_t size; /** book recordsize. This is a redundant field that helps in the implemen ta tion */
} indexbook;

typedef struct {
    indexbook *array;
    size_t used;
    size_t size;
} Array;
```

The struct **Book**, is used for storing book contents as the program reads input from the command line, it also simplifies the process of writing records into the data file.

The struct **Array,** is an static array passed as a pointer to add.c functions. Inside of it there is a maximum size “size” and “used” which is the number of records it currently contains. It also has a dynamic memory allocated array of structs type indexbook.

The struct **indexbook,** contains the information that is going to be written into the index file, plus the key, which is necessary to do the binary search.

```
/***delete**/
typedef struct {
    int key;  /*book id*/
    size_t registersize; /*size of the deleted register*/
    size_t offset; /*Record’s offset in file */
}indexdeletedbook ;

typedef struct {
    indexdeletedbook *array;
    size_t used;
    size_t size;
} Arraydel;
```

The struct **Arraydel** is exactly the same as Array structure but is used only for deleted records, and instead of an array of indexbooks it contains an array of indexdeletedbooks.

The struct **indexdelbook**, contains all the contents that are to be stored in the test.lst file and the key of each record, used for binary search and debugging purposes.

User interface:

To develop the library fully, we made a main function for the user interface (main.c) . The main function is in charge of:

1. calling functions to allocate memory for Array and Arraydel.

2. Creating test.db test.lst test.ind files according to the name passed through the command line.

3. Identifying what the command is. If it is add, it breaks the input into pieces considering that each field is separated by a "|", strtok is used for this, the information is stored in a type book variable and passed to function add_data, which stores it in the datafile, indexfile and Array.

4. freeing memory stored for Array, Arraydel and any other variable for which memory was reserved.

- All functions are implemented in add.c except for create_datafile, create_indexfile and create_listfile.
- After each command is executed the program prints *exit* in the console. For ending the program the user has to type *exit* so the loop ends.

Additional functions

There are some other functions we had to implement that where useful.

- **Sort_index**

  This function was used when adding a record, since indices must be ordered according to the key. It was also useful to sort the index again when a record was deleted.

- **Book *get_book(char *datafile, long int position, size_t size)**

  This function receives a position of a record in the datafile and looks for it, then it copies the whole record from the data file into a variable of type book, assigning the fields in book, the corresponding values. It was useful for function printRec function and for binary search.

- **void update_index(Array *index, int *inidpos)**

  When a record is deleted, this function is in charge of shifting all records so that the record that is deleted is no longer in the Array (index) variable.

Additional tests

Apart from the given tests, we implemented new ones to test certain functionalities:

Reload_index.sh: executes add_index_test.sh and then executes the program so it test that the index is correctly reloaded.

Reload_lst.sh: Reloading the list is necessary to know which gaps to fill when adding new records. This test executes add_delete_test_02.sh and then checks that the list of deleted records is correctly reloaded.

Add_delete_fill.sh: Tests that new records are added in empty spaces left by deleted records.