Goal

In this assignment, you are asked to implement a File Management System library in C.

Implementation Details & Requirements

File Management System (FMS) library has functionalities to handle a file, i.e., Create, Open, Close, Write, Delete, Find, FindNext, Re-index. FMS creates two files behind this system, which are data file (.dat), and indexing file (.ndx). Data file consists of two parts; *header* and *record.* Header part will be taken from "header.json" file by json parsing. Header part holds general information about file. "recordLength" is the size of a record. "keyStart" and "keyEnd" are the starting and ending bytes of key (inclusive in the end) that will be used for indexing. "order" holds the direction of the sorting (ASC: ascending, DESC: descending) . Besides, a new parameter called "open" must be added to header part, which represents the current status of the file (true=open and false=closed).

header.json file:

```
{ "recordLength" : 70, "keyStart" : 45 , "keyEnd" : 54, "order" : "ASC"}
```

An example record type for the given header.json:

```
struct RECORD {
        char team[44];
        char shortname[10];
        int rank;
        int goalscored;
        int goalconceeded;
        int point;
};
```

Data File Template:

	70|45|54|ASC|false
Offset: 18	Giresunspor|GRSN|2|30|3|28
Offset: 88	Hatayspor|HTY|3|28|5|25
Offset: 158	Beşiktaş|BJK|12|15|25|10
Offset: 228	Galatasaray|GS|10|19|20|14
Offset: 298	Fenerbahçe|FB|8|20|19|18
	...|...|...|...|...|...

Indexing File Template:

BJK|158
FB|298
GRSN|18
GS|228
HTY|88
...|...

.ndx index file consists of key/starting byte for each row of the record. In the above example, the key is placed between the bytes of 'keyStart' (45) and 'keyEnd' (54) which represents 'shortname' parameter of the record. According to the given scenario, header part stores 17 bytes. So, the first record (key = 'GRSN') starts from 18th byte of the memory. Considering this example, indexing file is the file holding key/starting byte, which is ordered according to key (shortname). Note: order is 'ASC' in this example.

TASKS

You need to implement functions defined in the following fms.h file.

```
 FILE *p;    // File Pointer (for data file)
 void *ndx;   //key-sort array (To hold data of indexing file)

 int myFileCreate(char * myfilename, char
 *jsonfilename); /**************
This function creates an empty file (data file) with the name given in the
first parameter "myfilename" and then the header of the respective file is
taken from the second parameter "jsonfilename". A variable called 'open'
should be initialized in the header part. It returns integer value 1 if
creation is successful, otherwise 0.
It also should create an empty file for indexing.
*****************/

int myFileOpen(char * myfilename);
 /**************
This function requires the name of the created file (data file). After
opening the file, 'open' parameter of the header will be changed to true.
There must be all the necessary controls such as "file not found", in case
file is not created. It returns integer value 1 if the file is successfully
opened, otherwise 0.
It should also open the indexing file and write all data into key-sort array
(ndx pointer). Then it closes indexing file.
*****************/

int myFileClose();
 /**************
This function doesn't require any parameter. It closes data file. Before
closing the data file, 'open' parameter of the header will be changed to
false and other necessary changes must be done such as re-indexing if
needed. It returns integer value 1 if the file is successfully closed,
otherwise 0.
*****************/

int myFileWrite(void *data);
 /**************
Insert the data into the file. It returns integer value 1 if the data is
successfully written, otherwise 0.
Note 1: The data will be added to data file but not to indexing file in this
part. It should update key-sort array (ndx pointer).
*****************/

int myFileDelete(void * key);
 /**************
Delete the data with the given key from the file. It returns integer value 1
if deletion is successful, otherwise 0.
*****************/

int myWriteIndex();
 /**************
This function writes all data from key-sort array (ndx pointer) into
indexing file. It returns integer value 1 if the data is successfully
written, otherwise 0.
*****************/

void printIndexingFile();
/** this function prints all elements in Indexing file **/

void printDataFile();
/** this function prints all records in Data file **/
```

```
int myFileFind(void * key, void * data);
 /**************
Find the data with the given key in the file. If the key is found in key-
sort array (ndx pointer), the parameter 'data' should be loaded with the
found data corresponding to the given key. It returns integer value 1 if key
is found, otherwise 0.
Note 2: Search operation should be done in key-sort array (ndx pointer) to
find the place of actual data in data file.
Note 3: This function should control if key-sort array (ndx pointer) is
updated.
*****************/

int myFileFindNext(void * key, void * data);
 /**************
Find the next key of the given key in the key-sort array (ndx pointer). If
key is found, the parameter 'data' should be loaded with the data
corresponding to the key. It returns integer value 1 if data is found,
otherwise 0.
Consider the same notes (2,3) for this function, too.
E.g.: if the input key is 'GRSN', it should return the data of 'GS'
*****************/

int myFileReindex();
 /**************
This function will be used in MyFileClose function, if any
insertion/deletion is done and key-sort array (ndx pointer) is not
written to indexing file. It returns integer value 1 if re-indexing is
successful, otherwise 0.
*****************/
 /**************
Note 4: You are free to add other global variables and other parameters as
input to the functions. However, return type will not change.
Note 5: This library will not be aware of your structures. Therefore, you
must use void pointers in this library. You can use casting in your .c file.
******/
```

How to create a library:

Create a file and name it as fms.h. Then write your all functions into fms.h. Now you are able to use this library as you use other libraries like xmllib,stdio etc.

example.h file

```
int carpma(int a, int b){
    return a * b;
}
```

test.c file

```
#include <stdio.h>
#include "example.h"
//fms.h and test.c should be in the same directory; otherwise, you need to
//write the path

int main(void)
{
  int sonuc;
  sonuc = carpma(5,6);
  printf("%d\n",sonuc);
}
```

You will run "gcc test.c"