LIBRASYNC DOCUMENTATION

VERSION: 1.0

GITHUB: https://github.com/krishna18developer/CampusConnect/

Librasync is a command-line application designed to automate library tasks. It facilitates book and patron management, borrowing, returning, and searching functionalities.

TEAM DETAILS:

1. KRISHNA TEJA MEKALA – 23EG109A34
2. KSHITIJ TIWARI – 23EG109A35
3. J. SANJANA – 23EG109A25
4. TARAKA SRINIVAS MERUGA – 23EG109A41
5. V. RASMISHA – 23EG109A65

THIS DOCUMENT CONTAINS CLEAR WORKING MECHANISM FOR EACH AND EVERY FUNCTION PRESENT IN THE LIBRASYNC – LIBRARY MANAGEMENT SYSTEM.

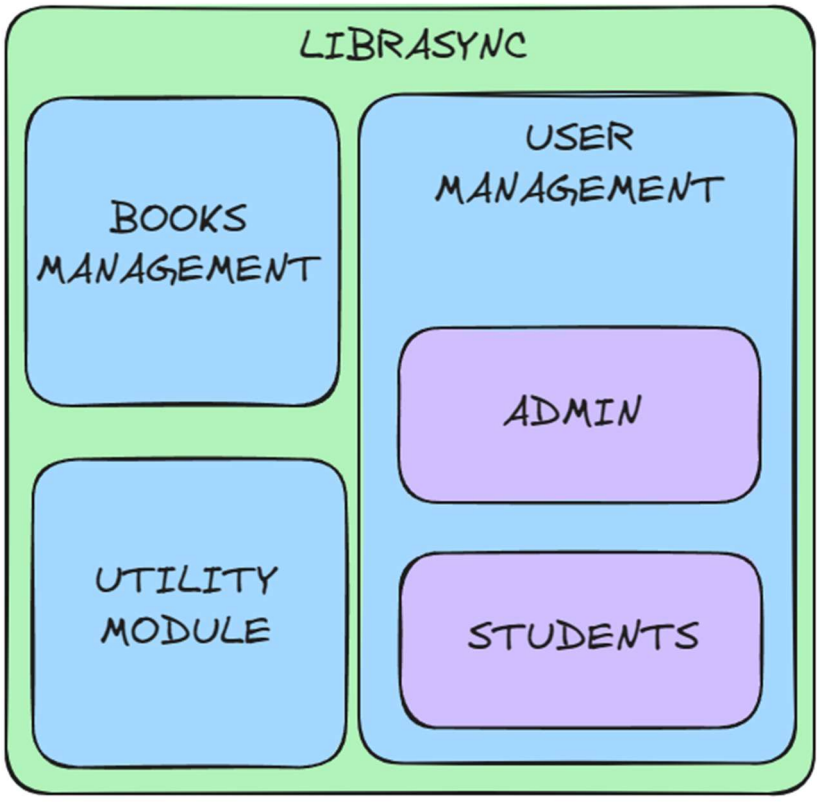
PROGRAMMING LANGUAGE: C (C99)

THE LIBRASYNC WILL HEREBY BE REFERENCED AS THE MAIN PROGRAM IN THE FOLLOWING PAGES.

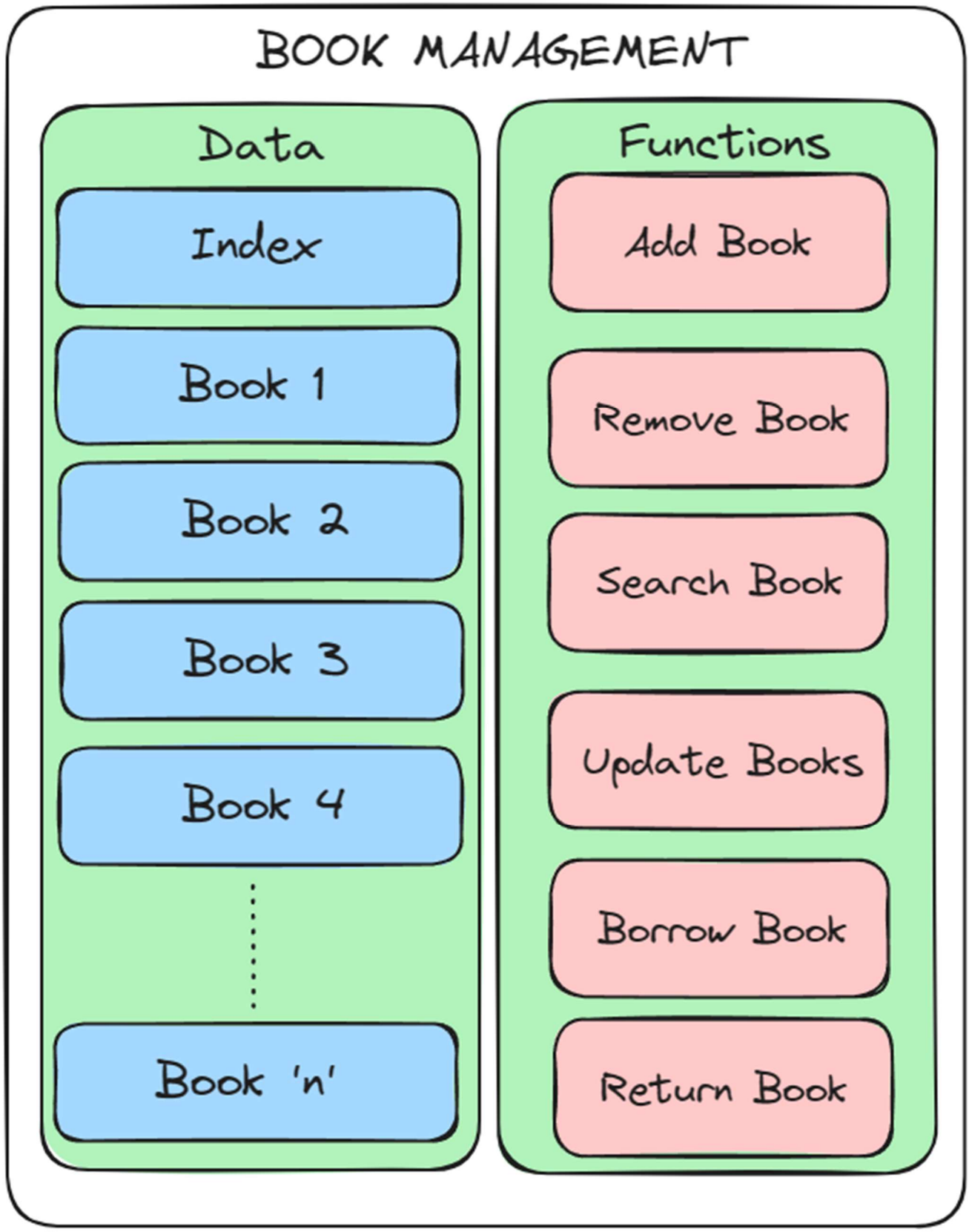
THE MAIN PROGRAM IS DIVIDED INTO 3 PARTS

1. BOOK MANAGEMENT
2. USER MANAGEMENT
3. UTILITY

EACH MODULE WILL BE COVERED IN THE FOLLOWING PAGES.



BOOK MANAGEMENT



DATA

INDEX

1. MORE PRECISELY " BINDEX.TXT " CONTAINS THE RELEVANT DATA ENTRY OF NUMBER OF BOOKS PRESENT IN THE MAIN PROGRAM
2. IT CONTAINS THE UID OF EACH AND EVERY BOOK.
3. THEREBY HELPS US RETRIEVE THE REFERENCE TO ALL THE BOOKS

BOOK

1. EACH BOOK FILE NAME IS AN UUID
2. EXAMPLE "B-b6037404-39d0-42e3-8a61-fa9f1b346625.txt"
3. EACH FILE WILL CONTAIN DETAILS OF THE BOOK SUCH AS
    a) UUID
    b) NAME
    c) AUTHOR
    d) GENRE
    e) PRICE
    f) PUBLISHED YEAR
    g) NUMBER OF COPIES
    h) NUMBER OF PEOPLE BORROWED
    i) BORROWED USERS

VARIABLES REQUIRED

```
Book* totalBooks, *newSetOfBooks;
Index* bIndex;
int numberOfBooks,numberOfNewBooks;
```

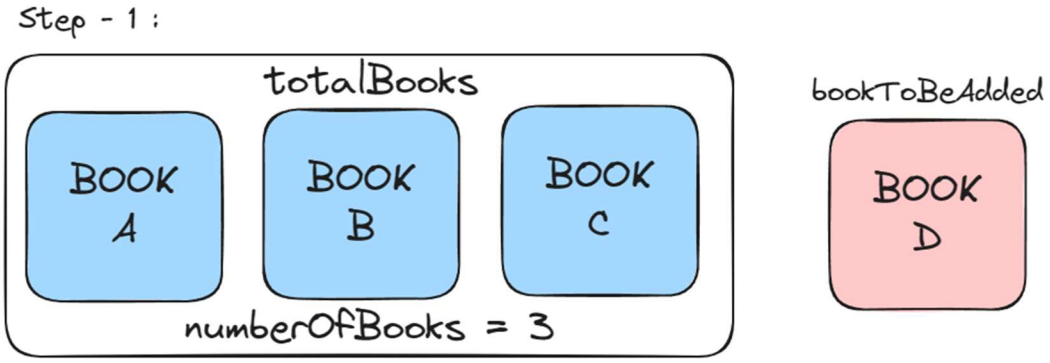
FUNCTIONS

ADD BOOK

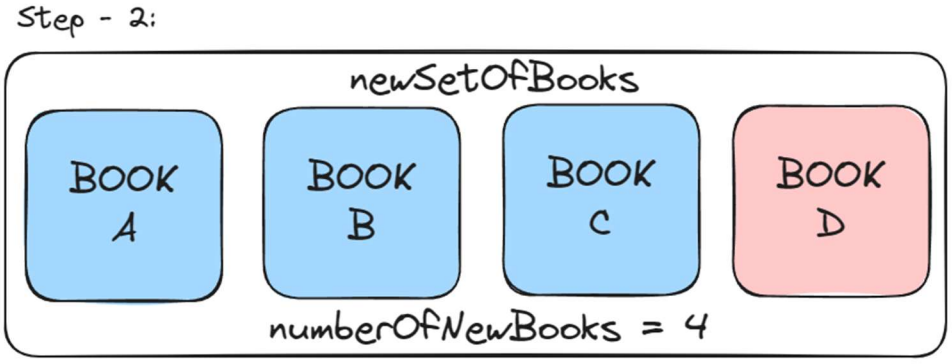
- THIS FUNCTION TAKES 1 PARAMETER, Book bookToBeAdded.

```
void AddBook(Book bookToBeAdded);
```

WORKING PRINCIPLE

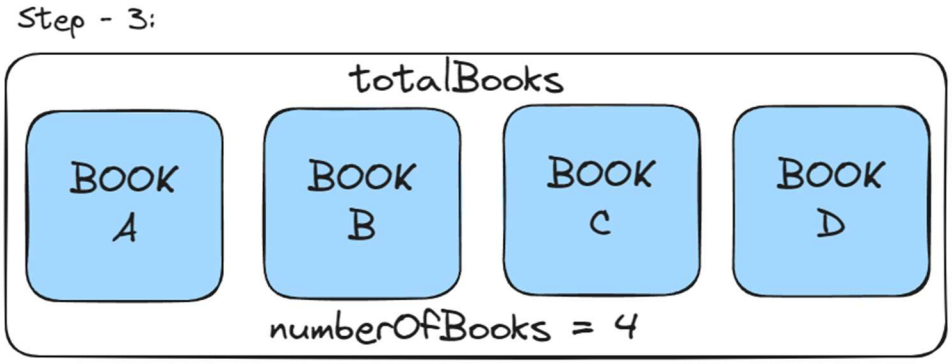


TAKE A NEW DYNAMICALLY ALLOCATED ARRAY OF SIZE numberOfNewBooks = numberOfBooks + 1



STORE ALL THE BOOKS FROM totalBooks IN newSetOfBooks, AND STORE THE bookToBeAdded AT THE index numberOfBooks, INCREMENT numberOfbooks BY 1. ADD THE UUID TO INDEX LIST.

FREE THE totalBooks MEMORY SPACE AND THEN EQUATE IT TO newSetOfBooks.

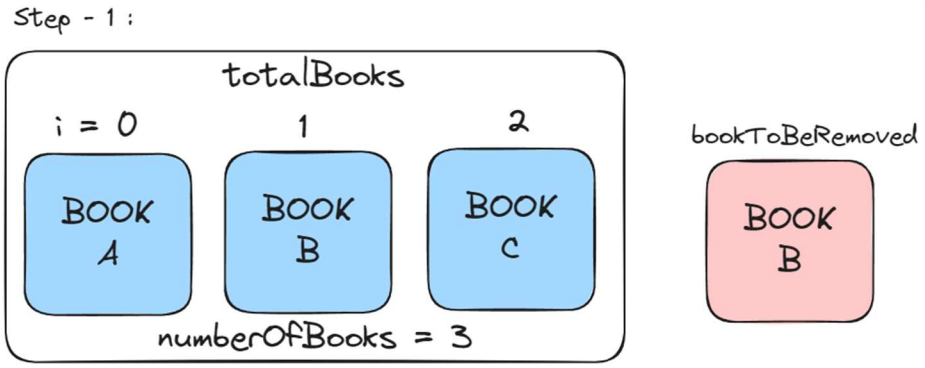


REMOVE BOOK

- THIS FUNCTION TAKES 1 PARAMETER, Book bookTobeRemoved

```
1  void RemoveBook(Book bookToBeRemoved);
```

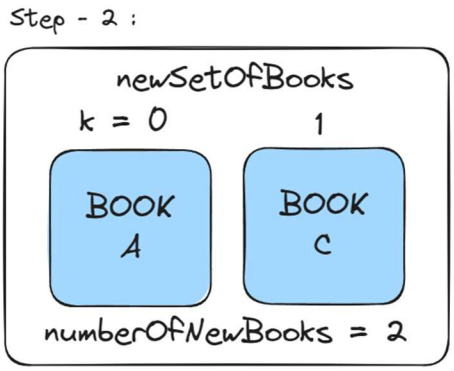
WORKING PRINCIPLE



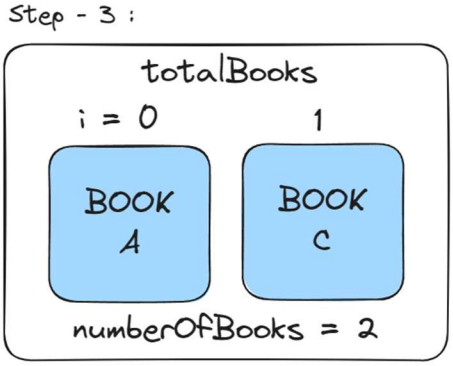
TAKE A NEW DYNAMICALLY ALLOCATED ARRAY OF SIZE numberOfNewBooks = numberOfBooks - 1

NOW COMPARE EACH BOOK IN totalBooks WITH bookToBeRemoved, IF THE BOOKS ARE NOT EQUAL THEN ADD IT INTO THE newSetOfBooks

DECREMENT numberOfbooks BY 1. REMOVE THE UUID FROM INDEX LIST.



FREE THE totalBooks MEMORY SPACE AND THEN EQUATE IT TO newSetOfBooks.



SEARCH BOOK

- THIS FUNCTION TAKES 1 PARAMETER, int searchType.

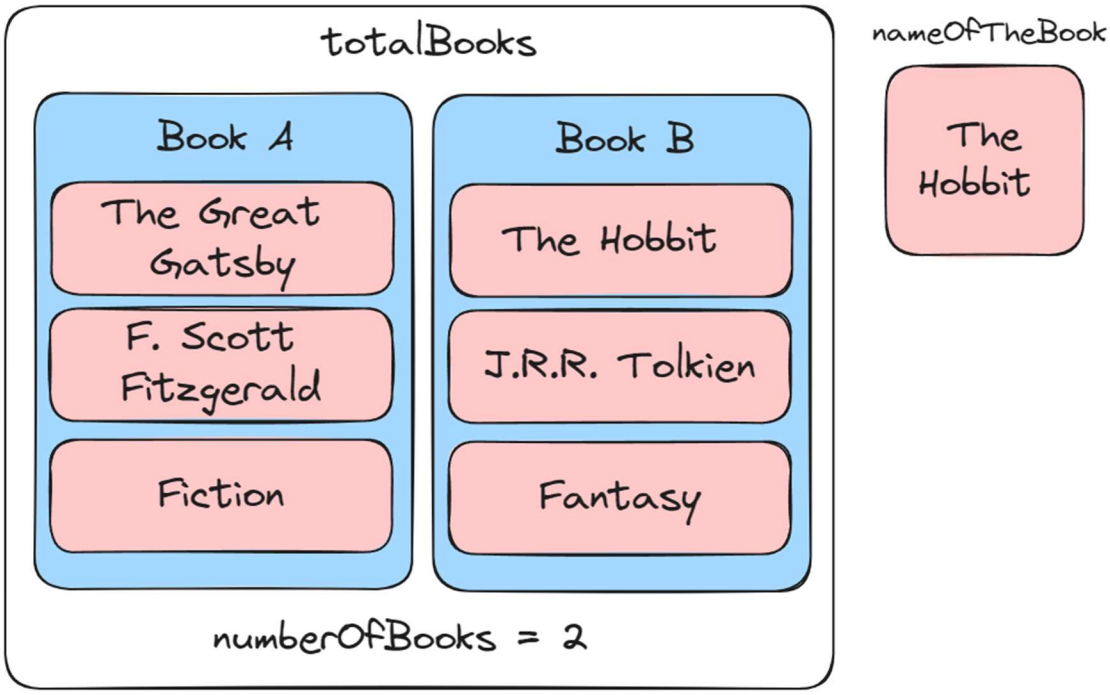
```
void SearchBook(int searchType);
```

searchType accepetable values are

BYNAME, BYAUTHOR, BYGENRE (REFER CONSTANTS.H)

WORKING PRINCIPLE

INTRODUCE A SWITCH CASE ON searchType AND IMPLEMENT APPROPRIATE LOGIC FOR SEARCHING OF BOOKS EITHER BY NAME, AUTHOR OR GENRE.



BUT HERE LIES GENERALISED WORKING PRINCIPLE FOR CHECKING.

IN THIS EXAMPLE, CHECKING BY NAME IS IMPLEMENT.

FIRST CHECK IF THE NAME OF BOOK IS EQUAL TO NAME IN EACH BOOK, IF TRUE PRINT THE BOOKS FOUND WITH MATCHING FACTOR.

UPDATE BOOKS

STORE THE LIST OF BOOK UUIDS IN INDEX FILE

THEN STORE EACH AND EVERY BOOK TO ITS INDIVIUDAL FILE

WITH ITS UUID NAME AND PREFIX “B-“

```
1  void UpdateBooks();
```

BORROW BOOK

STORE THE UUID OF USER BORROWING IN BOOK FILE

DECREMENT NUMBER OF COPIES

INCREMENT NUMBER OF BORROWED PEOPLE

```
1  void borrowBook(Book bookToBorrow);
```

RETURN BOOK

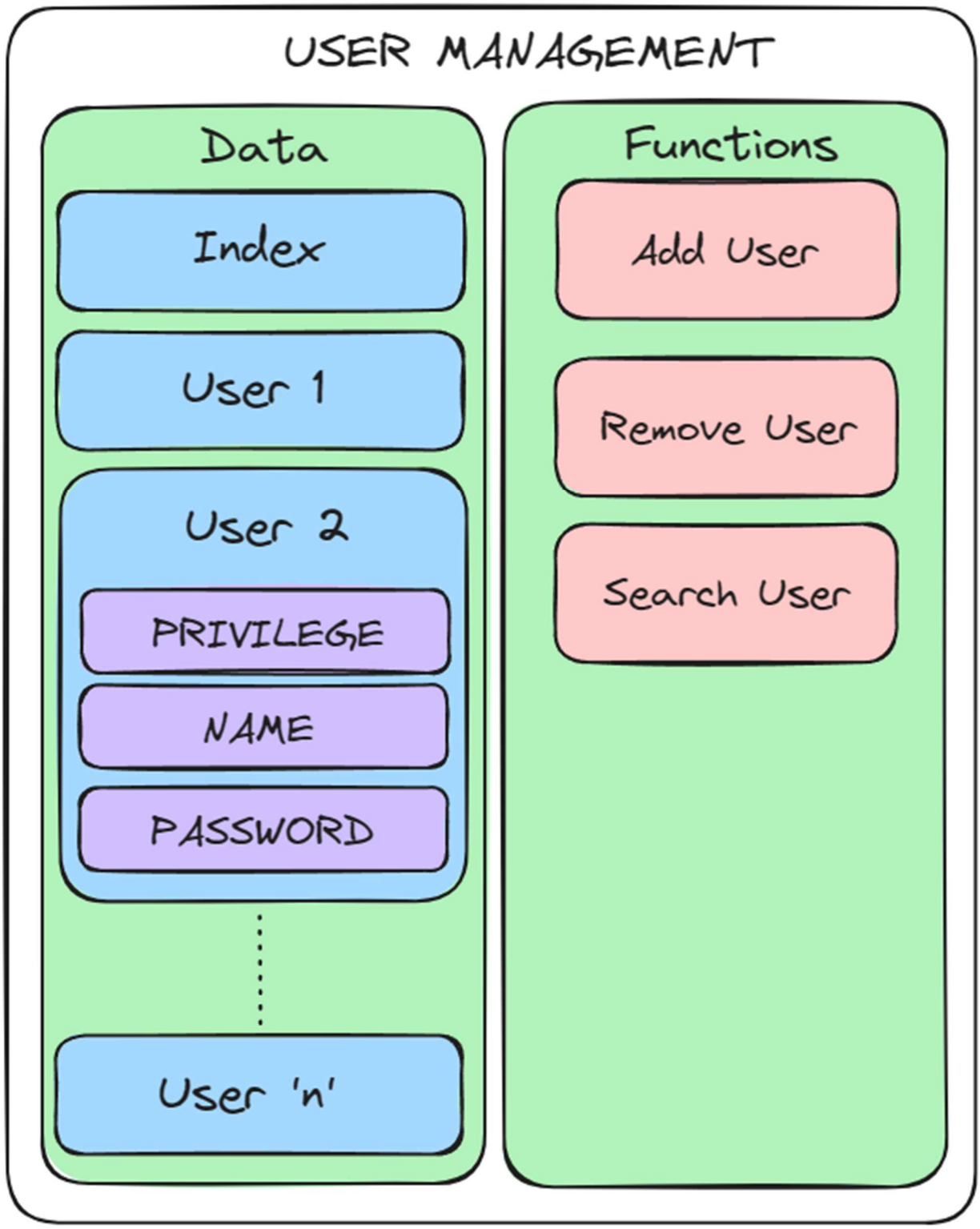
REMOVE THE UUID OF USER RETURN IN BOOK FILE

INCREMENT NUMBER OF COPIES

DECREMENT NUMBER OF BORROWED PEOPLE

```
1  void returnBook(Book bookToReturn);
```

USER MANAGEMENT



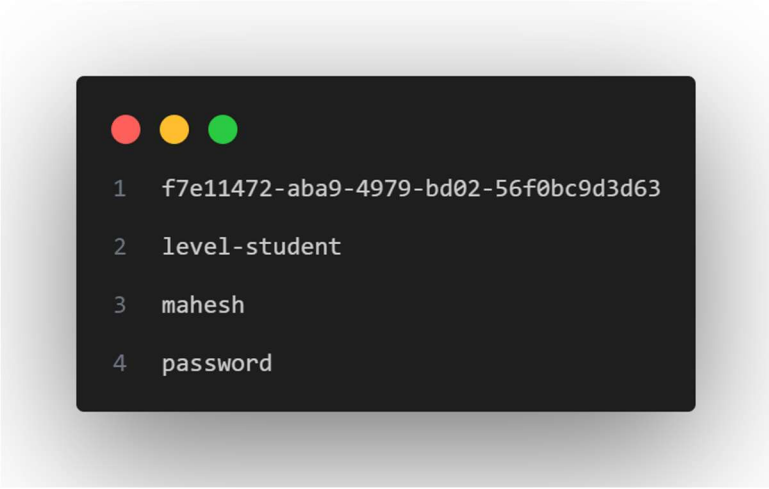
DATA

INDEX

4. MORE PRECISELY " UINDEX.TXT " CONTAINS THE RELEVANT DATA ENTRY OF NUMBER OF USERS PRESENT IN THE MAIN PROGRAM
5. IT CONTAINS THE UID OF EACH AND EVERY USER.
6. THEREBY HELPS US RETRIEVE THE REFERENCE TO ALL THE USERS

USER

1. EACH USER FILE NAME IS AN UUID
2. EXAMPLE "U-f7e11472-aba9-4979-bd02-56f0bc9d3d63.txt"
3. EACH FILE WILL CONTAIN DETAILS OF THE USER SUCH AS
    a) UUID
    b) PRIVILEGE
    c) NAME
    d) PASSWORD



VARIABLES REQUIRED

```
Book* totalBooks, *newSetOfBooks;
Index* bIndex;
int numberOfBooks,numberOfNewBooks;
```

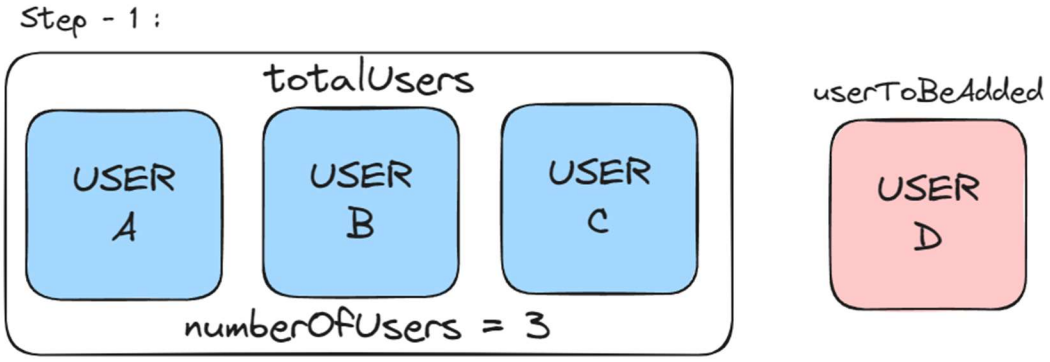
FUNCTIONS

ADD USER

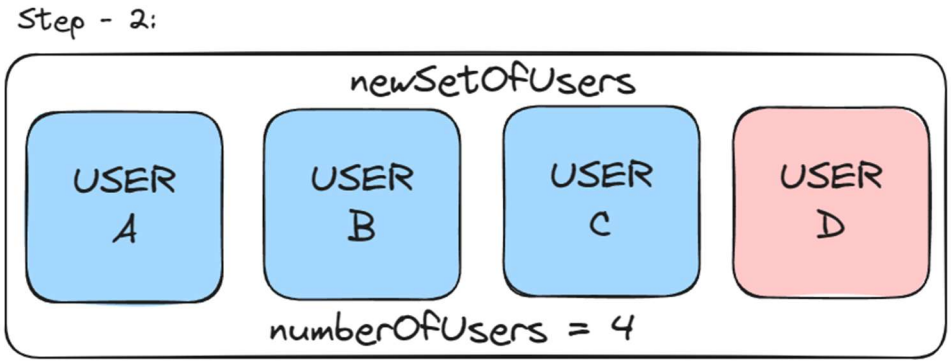
- THIS FUNCTION TAKES 1 PARAMETER, User userToBeAdded.

```
1  void AddUser(User userToBeAdded);
```

WORKING PRINCIPLE

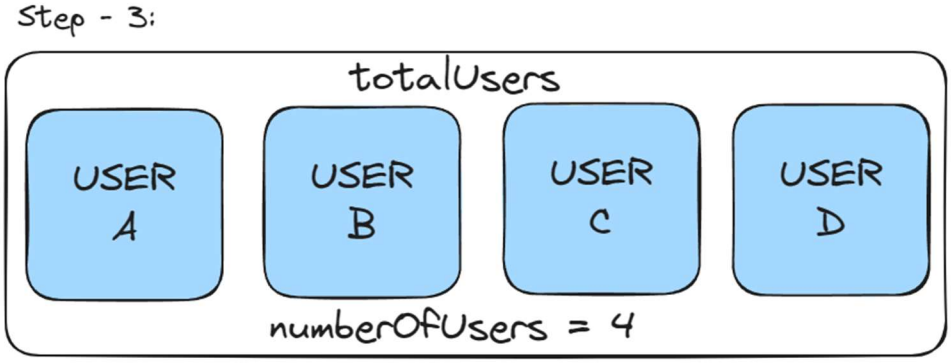


TAKE A NEW DYNAMICALLY ALLOCATED ARRAY OF SIZE numberOfNewUsers = numberOfUsers + 1



STORE ALL THE USERS FROM totalUsers IN newSetOfUsers, AND STORE THE userToBeAdded AT THE index numberOfUsers, INCREMENT numberOfUsers BY 1. ADD THE UUID TO INDEX LIST.

FREE THE totalUsers MEMORY SPACE AND THEN EQUATE IT TO newSetOfUsers.

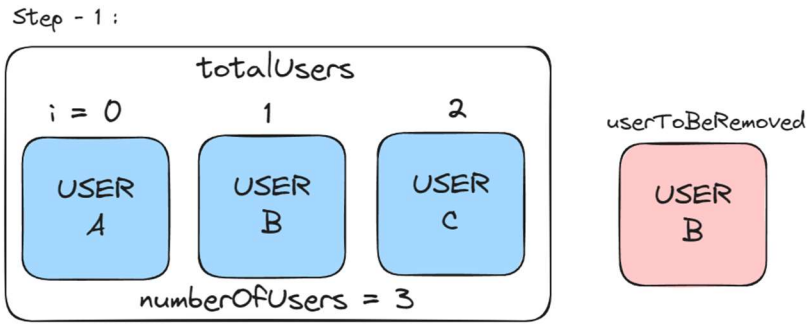


REMOVE USER

- THIS FUNCTION TAKES 1 PARAMETER, User userTobeRemoved

```
1  void RemoveUser(User userToBeRemoved);
```

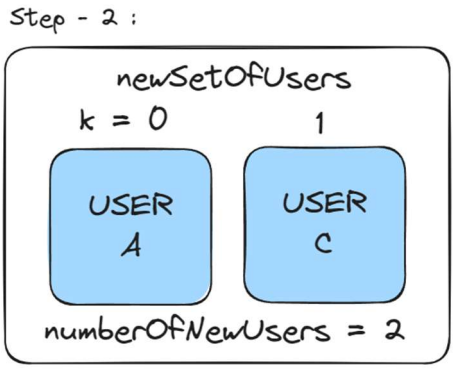
WORKING PRINCIPLE



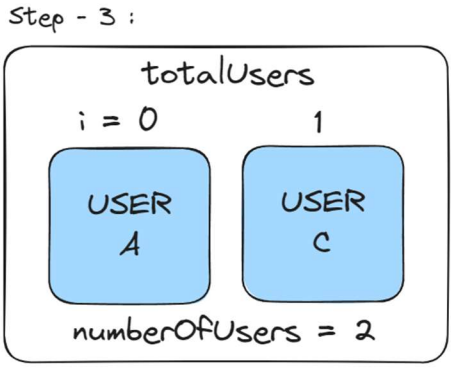
TAKE A NEW DYNAMICALLY ALLOCATED ARRAY OF SIZE numberOfNewUsers = numberOfUsers - 1

NOW COMPARE EACH USER IN totalUsers WITH usersToBeRemoved, IF THE USERS ARE NOT EQUAL THEN ADD IT INTO THE newSetOUsers

DECREMENT numberOfUsers BY 1. REMOVE THE UUID FROM INDEX LIST.



FREE THE totalUsers MEMORY SPACE AND THEN EQUATE IT TO newSetOUsers.



SEARCH USER

- THIS FUNCTION TAKES 1 PARAMETER, int searchType.

```
User SearchUser(User userToSearch);
```

searchType accepetable values are

BYNAME, BYAUTHOR, BYGENRE (REFER CONSTANTS.H)

WORKING PRINCIPLE

INTRODUCE A SWITCH CASE ON searchType AND IMPLEMENT APPROPRIATE LOGIC FOR SEARCHING OF USER EITHER BY NAME OR UUID.

BUT HERE LIES GENERALISED WORKING PRINCIPLE FOR CHECKING.

IN THIS EXAMPLE, CHECKING BY NAME IS IMPLEMENT.

FIRST CHECK IF THE NAME OF USER IS EQUAL TO NAME IN EACH USER, IF TRUE PRINT THE USER FOUND WITH MATCHING FACTOR.

UPDATE USER

STORE THE LIST OF USER UUIDS IN INDEX FILE

THEN STORE EACH AND EVERY USER TO ITS INDIVIUDAL FILE

WITH ITS UUID NAME AND PREFIX “U-“

```
void UpdateBooks();
```