LAB 2 - (1) Qns - Single Linked Lists

Single Linked Lists

Emma and Olivia are avid readers who have two small personal bookshelves to store their book collections. They decide to merge their book collections into a new combined bookshelf while following these rules:

1) The books on each shelf are already arranged in alphabetical order. 2) When combining the collections, they will keep books with the same titles on the new shelf and discard duplicates from Olivia’s collection. 3) The common bookshelf should maintain the alphabetical order of book titles across the shelf.

The bookshelves are represented by Singly Linked Lists with the books arranged in the given order. NO NEW BOOKS ARE ALLOWED to be a part of the merged collection.

Input Format

1st line - 2 integers m n m - number of books in Emma’s bookshelf n - number of books in Olivia’s bookshelf 2nd line - m names of books in Emma’s bookshelf in sorted order 3rd line - n names of books in Olivia’s bookshelf in sorted order

Constraints

0<=m<=1000 0<=n<=1000

Output Format

Single line representing the names of books in the new bookshelf in sorted order with ‘ - ‘ in between the book names.

Sample Input 0

```
3 4
HarryPotter LordOfTheRings TheHobbit
Divergent FiftyShades HungerGames Twilight
```

Sample Output 0

```
Divergent - FiftyShades - HarryPotter - HungerGames - LordOfTheRings - TheHobbit - Twilight -
```

💡 answer :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct BookNode
{
    char title[100];
    struct BookNode *next;
} BookNode;

BookNode *createBook(const char *bookTitle)
{
    BookNode *newBook = (BookNode *)malloc(sizeof(BookNode));
    strcpy(newBook->title, bookTitle);
    newBook->next = NULL;
    return newBook;
}

BookNode *mergeBookshelves(BookNode *shelf1, BookNode *shelf2)
{
    BookNode *shelf=shelf1;
    while(shelf->next!=NULL)
    {
        shelf=shelf->next;
    }
    shelf->next=shelf2;

    if (shelf1 == NULL || shelf1->next == NULL) {
        return shelf1;
    }
    int swapped;
    BookNode* current;
    BookNode* last = NULL;
    do {
        swapped = 0;
        current = shelf1;
        while (current->next != last) {
            if (strcmp(current->title, current->next->title)>0) {
                char temp[100];
                strcpy(temp,current->title);
                strcpy(current->title,current->next->title);
                strcpy(current->next->title,temp);
                swapped = 1;
            }
            current = current->next;
        }
        last = current;
    } while (swapped);

      if (!shelf1) {
```

```
        return NULL;
    }

    BookNode* current2 = shelf1;
    BookNode* prev2 = NULL;
    while (current2) {
        BookNode* inner_current = current2->next;
        BookNode* inner_prev = current2;

        while (inner_current) {
            if (strcmp(current2->title, inner_current->title) == 0) {
                inner_prev->next = inner_current->next;
                free(inner_current);
                inner_current = inner_prev->next;
            } else {
                inner_prev = inner_current;
                inner_current = inner_current->next;
            }
        }

        prev2 = current2;
        current2 = current2->next;
    }

    return shelf1;
}

void displayMergedBookshelf(BookNode *start)
{
    BookNode *current = start;
    while (current)
    {
        printf("%s - ", current->title);
        current = current->next;
    }
}


int main()
{
    int emmaCount, oliviaCount;
    scanf("%d %d", &emmaCount, &oliviaCount);
    BookNode *emmaShelf = NULL, *oliviaShelf = NULL, *temp = NULL;
    char bookTitle[100];

    if (emmaCount > 0)
    {
        scanf("%s", bookTitle);
        emmaShelf = createBook(bookTitle);
        temp = emmaShelf;
        for (int i = 1; i < emmaCount; i++)
        {
            scanf("%s", bookTitle);
            temp->next = createBook(bookTitle);
            temp = temp->next;
        }
```

```
    }
    if (oliviaCount > 0)
    {
        scanf("%s", bookTitle);
        oliviaShelf = createBook(bookTitle);
        temp = oliviaShelf;
        for (int i = 1; i < oliviaCount; i++)
        {
            scanf("%s", bookTitle);
            temp->next = createBook(bookTitle);
            temp = temp->next;
        }
    }

    BookNode *mergedBookshelf = mergeBookshelves(emmaShelf, oliviaShelf);
    displayMergedBookshelf(mergedBookshelf);
    return 0;
}
```