```
/*
* Author: Linh Phan
* Course: COP3502
* Assignment: 2 Library Functions
* Date: 2/28/2025
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_TITLE 100
#define MAX_NAME 50

typedef struct node_t {
      char bookTitle[100];
      char lastName [50];
      char firstName[50];
      struct node_t *nextptr;
} node_t;

// Function prototypes
node_t *borrowBook(node_t *head, char *title, char *lastName, char *firstName, FILE
*output);
node_t *returnBook(node_t *head, char *title, char *lastName, char *firstName, FILE
*output);
void checkBook(node_t *head, char *title, char *lastName, char *firstName, FILE
*output);
void displayBorrowedBooks(node_t *head, FILE *output);
void freeList(node_t *head, FILE *output);

//memory allocation for nodes
node_t* createnode_t(char* bookTitle, char* lastName, char* firstName) {
      node_t* newnode = (node_t*)malloc(sizeof(node_t));
      if (newnode == NULL) {
            printf("Memory allocation failed\n");
            return NULL;
      }
      strcpy(newnode->bookTitle, bookTitle);
      strcpy(newnode->lastName, lastName);
      strcpy(newnode->firstName, firstName);
      newnode->nextptr = NULL;
      return newnode;
}

//attach it to a node in linked list
node_t* appendnode_t(node_t* head, char* bookTitle, char* lastName, char*
firstName) {
      node_t* newnode = createnode_t(bookTitle, lastName, firstName);
      if (head == NULL)
            return newnode;
      node_t* temp = head;
      while (temp->nextptr != NULL) {
            temp = temp->nextptr;
      }
      temp->nextptr = newnode;
      return head;
}

//delete a node
node_t* removeNode(node_t* head, char* bookTitle, char* lastName, char* firstName)
```

```
{
	node_t* temp = head, *prev = NULL;

//tranversing list to find the book that matches
	while (temp != NULL) {
		if (strcmp(temp->bookTitle, bookTitle) == 0 && strcmp(temp->lastName,
lastName) == 0 && strcmp(temp->firstName, firstName) == 0) {
			if (prev == NULL) {
				head = temp->nextptr;
			} else {
				prev->nextptr = temp->nextptr;
			}
			free(temp);
			return head;
		}
		prev = temp;
		temp = temp->nextptr;
	}
	return head;
}

node_t *borrowBook(node_t *head, char *title, char *lastName, char *firstName, FILE
*output) {
	head = appendnode_t(head, title, lastName, firstName);
	fprintf(output, "Borrowed \"%s\" by %s, %s\n", title, lastName, firstName);
	return head;
}

node_t *returnBook(node_t *head, char *title, char *lastName, char *firstName, FILE
*output) {
	fprintf(output, "Returned \"%s\" by %s, %s\n", title, lastName, firstName);
	head = removeNode(head, title, lastName, firstName);
	return head;
}


void checkBook(node_t *head, char *title, char *lastName, char *firstName, FILE
*output) {
	node_t* temp = head;
	//tranversing until we find book
	while (temp != NULL && (strcmp(temp->bookTitle, title) != 0) && (strcmp(temp-
>lastName, lastName) != 0) && (strcmp(temp->firstName, firstName))) {
		temp = temp->nextptr;
	}
	if (temp == NULL) {
		fprintf(output, "\"%s\" is not currently borrowed by %s, %s\n", title,
lastName, firstName);
		return;
	}
	else {
		fprintf(output, "\"%s\" is borrowed by %s, %s\n", title, lastName,
firstName);
		return;
	}
}

void displayBorrowedBooks(node_t *head, FILE *output) {
	fprintf(output, "Borrowed Books List: \n");
	int index = 0;
```

```
      node_t* temp = head;
      while (temp != NULL) {
            index ++;
            fprintf(output, "%d. \"%s\" - %s, %s\n", index, temp->bookTitle, temp-
>lastName, temp->firstName);
            temp = temp->nextptr;
      }
}

void freeList(node_t *head, FILE *output) {
      while (head != NULL) {
    node_t* temp = head;
            head = temp->nextptr;
            free(temp);
      }
}

int main (int argc, char* argv[]) {
    //opening files
      if (argc < 3) {
            printf("Usage: %s filename\n", argv[0]);
            return 1;
      }
      FILE *file = fopen(argv[1], "r");
      if (file == NULL) {
            printf("Error opening file: %s\n", argv[1]);
            return 1;
      }
      FILE *file2 = fopen(argv[2], "w");
      if (file2 == NULL) {
            printf("Error opening file: %s\n", argv[2]);
            return 1;
      }

      char bookTitle[100];
      char lastName [50];
      char firstName[50];
      node_t* availList= NULL, *borrowList = NULL;
      int option;

//scanning first number to get what option they want before doing the required
function
      while (fscanf(file, "%d", &option) == 1 && option != 5) {
            if (option == 1) {
                  if(fscanf(file, " \"%[^\"]\" %s %s", bookTitle, lastName,
firstName) == 3) {
                        borrowList = borrowBook(borrowList, bookTitle, lastName,
firstName, file2);
                        removeNode(availList, bookTitle, lastName, firstName);
                  }
            }
            else if (option == 2) {
                  if(fscanf(file, " \"%[^\"]\" %s %s", bookTitle, lastName,
firstName) == 3) {
                        appendnode_t(availList, bookTitle, lastName, firstName);
                        borrowList = returnBook(borrowList, bookTitle, lastName,
firstName, file2);
                  }
            }
```

```
            else if (option == 3) {
                  if(fscanf(file, " \"%[^\"]\" %s %s", bookTitle, lastName,
firstName) == 3) {
                        checkBook(borrowList, bookTitle, lastName, firstName,
file2);
                  }
            }
            else if (option == 4) {
                  displayBorrowedBooks(borrowList, file2);
            }
      }

      fprintf(file2, "Thank you for using the Library System!");

      fclose(file);
      fclose(file2);

      freeList(borrowList, file2);
      freeList(availList, file2);

      return 0;
}
```