Progetto Laboratorio Algoritmi e Strutture Dati

Nome: Antonio
Cognome: Petrillo
Matricola: N86002818

Considerazioni iniziali:
Il progetto richiede di realizzare un semplice programma per la gestione di una libreria, l'obiettivo principale è mostrare di avere dimestichezza con le strutture dati viste fino ad ora nel corso.
La libreria deve poter accogliere al suo interno 15 libri e gli studenti possono o restituire un libro o prenderne in prestito uno, inoltre le richieste possono essere gestite subito oppure sospese e terminate successivamente.
Per rappresentare "gli attori in scena", ovvero i libri e gli studenti ho realizzato due file separati, rispettivamente book.c e student.c (con annessi header .h), nei quali ho dichiarato tipi e funzioni per rappresentarli, ma non contengono nessuna struttura dati in particolare.

Specifiche dell'implementazione:
Per rappresentare la libreria in se, ovvero una collezione di libri, occorre una struttura dati dinamica dato che il prestito o la restituzione possono far variare la cardinalità dell'insieme, ma specificherò meglio a breve.
La mia scelta è ricaduta sugli alberi Red-Black.
Per gestire solamente 15 libri una struttura del genere è probabilmente inutile, ma l'obiettivo del corso è imparare appunto l'uso di strutture dati ed inoltre realizzare una semplice lista linkata o un array mi sembrava un po' troppo riduttivo.
L'uso specifico dei Red-Black è un puro esercizio di stile.
I vari nodi vengono comparati in base all'ordinamento lessicografico sfruttando l'ISBN dei libri.
Ogni nodo dell'albero contiene, oltre alle sue peculiarità (figlio dx, sx, padre e colore) contiene una variabile di tipo libro e un intero utilizzato per rappresentare il numero di copie di uno specifico libro.
Si potrebbe osservare quindi che anche se utilizzo una struttura dinamica essa non varia con l'aggiunta/prelievo di libri, ma cambia solamente il valore che specifica le copie presenti di un singolo libro in un singolo nodo preesistente (questo se consideriamo un albero già inizializzato), ed infatti è cosi, ma ho effettuato questa scelta per semplificare alcuni passaggi.
Infatti in una prima "versione" del progetto, se così la posso chiamare, avevo aggiunto ad ogni nodo dell'albero una lista linkata di libri , cosi da poter raggruppare insieme tutte le copie uguali di un libro ed ottenere alcuni vantaggi (oltre al fatto che era un'ottima scusa per usare un'ulteriore struttura dati), purtroppo il risultato finale era un sorgente di circa 1200+ righe di codice ed essendo questa solo una parte del progetto temevo di realizzare qualcosa di troppo sproporzionato e difficile da correggere ed ho optato per un refactoring del codice.
Una soluzione alternativa sarebbe potuta essere quella di inserire un nodo diverso per ogni copia di un libro, ma semanticamente la considero sbagliata o quantomeno fuorviante (in una biblioteca i libri uguali sono raggruppati insieme e posti vicini piuttosto che essere sparsi su vari ripiani).

Per la seconda parte delle richieste ho realizzato una coda, sempre dinamica (che con molta fantasia ho chiamato “request_queue”), nella quale poter aggiungere, nel caso non voglia eseguire subito un’operazione, sia richieste di prestito, che di restituzione.
In questo modo per riprendere le operazioni sospese basta effettuare un dequeue analizzare il contenuto del nodo rimosso ed eseguire l’operazione contenuta in esso.
L’utilizzo della coda è semanticamente corretto dato che la prima istruzione da completare deve essere la prima lasciata in sospeso, ovvero occorre seguire un politica FIFO.
Se una richiesta di prestito non può essere soddisfatta per l’assenza di copie essa viene aggiunta alla coda delle richieste.
Se si tenta di restituire un libro non appartenente alla libreria la richiesta viene rifiutata.
Infine la parte finale, il main, sul quale in realtà non c’è molto da dire si occupa solo di caricare i libri nell’albero, inizializzare la coda e gestire effettivamente le richieste.
In particolare i libri da caricare in memoria vengono letti da file, anche perché inserirli da tastiera ogni volta sarebbe stato un incubo ed inserirli direttamente nel codice è anche peggio.
Per poter leggere da file i libri occorre mettere in un determinato ordine i dati sul file, in particolare :

1) ISBN del libro;
2) titolo del libro;
3) numero di autori;
4.1) autore 1;
…
4.i) i-esimo autore;
…
4.n) n-esimo autore;
5) prezzo del libro;
6) numero copie.

Il parametro “numero di copie” serve per poter allocare un matrice di caratteri (un array di array di caratteri, in pratica un char**) .
Numero di copie rappresenta il numero di righe, le colonne vengono allocate successivamente in base alle varie stringhe.
La lunghezza massima possibile per una stringa è 255 (+ 1 per il terminatore ‘\0’) caratteri.
Il codice del main è leggermente sproporzionato per quel che fa. Ciò è dovuto alla gestione degli input da tastiera, in particolare per raccogliere un’intera linea, compresa di spazi, è leggermente macchinoso perché occorre ripulire i buffer dopo ogni lettura, rimuovere il carattere di newline da ogni stringa, ecc...
Infine ho aggiunto una funzione per creare un backup del file di input, che può tornare sempre utile, ed un’altra per riporre il contenuto della libreria in un file di output.

Esempio di esecuzione:
################### START PROGRAM ###################

################### LIBRARY LOADED ###################

################### BACKUP CREATED ###################

Enter a choice:
1) Add a return request
2) Add a loan request
3) Resume suspended requests
4) Print library content
5) End program
Any other choice will be ignored

choice: 2
insert student's name: Antonio
insert student's surname: Petrillo
insert student's matricola (9 char): N86002818
insert isbn (max 13 char): 9788806227920
insert title: Racconti di Pietroburgo
insert authors number: 1
insert the full name of the 1-ith author: Nikolaj Gogol'
insert the price of the book: 11.0

Do you want to handle the request now? [Y/n]: n

Enter a choice:
1) Add a return request
2) Add a loan request
3) Resume suspended requests
4) Print library content
5) End program
Any other choice will be ignored

choice: 2
insert student's name: DOOM
insert student's surname: Slayer
insert student's matricola (9 char): N86006669
insert isbn (max 13 char): 9788806227920
insert title: Racconti di Pietroburgo
insert authors number: 1
insert the full name of the 1-ith author: Nikolaj Gogol'
insert the price of the book: 11

Do you want to handle the request now? [Y/n]: y
matricola: N86006669, name: DOOM, surname: Slayer, requested requires the book:
ISBN: 9788806227920
title: Racconti di Pietroburgo
num_author: 1
authors:---
	|--> Nikolaj Gogol'
price: 11.000000

Enter a choice:
1) Add a return request
2) Add a loan request
3) Resume suspended requests
4) Print library content
5) End program
Any other choice will be ignored

choice: 3
there are no copies available for (isbn) 9788806227920
matricola: N86002818, name: Antonio, surname: Petrillo, requested requires the book:
ISBN: 9788806227920
title: Racconti di Pietroburgo
num_author: 1
authors:---
	|--> Nikolaj Gogol'
price: 11.000000

book not present in the library, request suspended

Enter a choice:
1) Add a return request
2) Add a loan request
3) Resume suspended requests
4) Print library content
5) End program
Any other choice will be ignored

choice: 1
insert student's name: DOOM
insert student's surname: Icorn
insert student's matricola (9 char): N86009666
insert isbn (max 13 char): 9788806227920
insert title: Racconti di Pietroburgo
insert authors number: 1
insert the full name of the 1-ith author: Nikolaj Gogol'
insert the price of the book: 11.0

Do you want to handle the request now? [Y/n]: y
matricola: N86009666, name: DOOM, surname: Icorn, returned the book:
ISBN: 9788806227920
title: Racconti di Pietroburgo
num_author: 1
authors:---
	|--> Nikolaj Gogol'
price: 11.000000

Enter a choice:
1) Add a return request
2) Add a loan request
3) Resume suspended requests
4) Print library content
5) End program
Any other choice will be ignored

choice: 3
matricola: N86002818, name: Antonio, surname: Petrillo, requested requires the book:
ISBN: 9788806227920
title: Racconti di Pietroburgo
num_author: 1
authors:---
	|--> Nikolaj Gogol'
price: 11.000000

Enter a choice:
1) Add a return request
2) Add a loan request
3) Resume suspended requests
4) Print library content
5) End program
Any other choice will be ignored

choice: 5
##################### END PROGRAM ####################

Conclusione:
Per compilare il progetto ho utilizzato il compilatore gcc, con il seguente comando:

gcc main.c rbtree.c request_queue.c student.c book.c -o progetto1

Ho lanciato l'eseguibile sfruttando valgrind (è un programma che permette di vedere la gestione della memoria da parte del software) e non ho trovato memory leak.
Per eseguire il programma occorre passare come primo argomento il file da cui leggere la libreria (ed il file deve essere formattata come scritto di sopra).
Ho allegato anche un altro file in cui descrivo più nel dettaglio come ho realizzato gli alberi Red-Black.

Allego anche il link a github per scaricare tutti i file del progetto:
https://github.com/antonio-petrillo/laboratorio-algoritmi-e-strutture-dati/tree/master/progetto1