ECE 36800 – Data Structures
Programming Assignment 2 – Part 1

Purpose of Assignment 2:
Design and implement a bookstore user interactive program.

Goal of Part 1:
Get familiar with some basic linked list processing and set up necessary components in preparation for part 2.

What to submit:

1. `booklist1.cpp`: The implementation file for the new booklist class. Start with the given file and add your implementation.
2. `catalog.cpp`: The application program displaying the bookstore catalog as well as searching by id functionality. Start with the given file and add your implementation.
3. A word document `proj2p1.docx`: This file should include the printout of your program. For testing of `booklist::search_by_id` function, please make sure you have sufficient test cases to show the correctness of your program. Include a copy (or screenshot) of the printout of the program run for each test case.
4. Push all your files under the "proj2/part1" directory before the deadline.

Other available files (please do not modify):

1. `book.h` and `book.cpp`: The implementation file for the book class. Please note that the `<iomanip>` header is included to print the output according to certain format.
2. `booklist1.h`: The header file for the new booklist class, which is implemented using a singly linked list.
3. `booklist.txt`: A list of books along with their features, each delimited by a TAB.
4. `Makefile`: rules to compile the source files for your convenience.

Guideline:

Once starts, the program will read from a book catalog file, called `booklist.txt`, with a list of books and their features as in Table 1.

Table 1: A sample bookstore catalog.

ID	Title	Price	Copies
1	Lisey's Story	$28.00	10
2	The March	$14.95	10
3	The Lincoln Lawyer	$26.95	10
4	The Morning Star	$17.00	10
5	Tuesdays with Morrie	$6.99	15
6	The Da Vinci Code	$7.99	15
7	The Rings Book	$24.95	12
8	Following in Lincoln's Footsteps	$16.00	8
9	The Lord of the Rings	$35.00	10
10	The Perfect Storm	$14.00	12

After reading from `booklist.txt`, the program stores the books in a linked list called `catalog`, then prints all the books in `catalog`. Next, the program will test the `booklist::search_by_id` function by asking the user to enter a book id. If a specified book id is found in the catalog, the information about the book will be printed out. In the end, the program will test the `booklist::remove_book` function by asking the user to enter a book id. If a specified book id is found, the book will be removed from the catalog, and the updated catalog will be printed out as well. the information about the book will be print out.

Explanations of Implementation:

- The `booklist` is a singly-linked list where each of its node holds a book pointer as well as a node pointer to the next node in the list. There is a `head` member variable that gives access to the first node in the booklist. `catalog` is defined as a `booklist` object.
- In the `fill_catalog` function in catalog.cpp, each time after reading a new line from `booklist.txt`, a book object is created (dynamically through the **new** keyword). To add these books to the catalog, the `booklist::add_book` function is called, which first create a node object (again dynamically through the **new** keyword), then this new node will be appended to the back of the current `catalog` booklist. The memory spaces of these node objects and book objects are released later through the **delete** keyword - as shown in the destructor of `booklist` and the destructor of `book`.
- The `booklist::remove_book` function should be able to handle the following different cases appropriately: 1. If the book with the specified id is not found, then the booklist is not changed; 2. If the book with the specified id is found, then the book node will be removed from the list (through the **new** keyword); In addition, if the book node is in the front of the booklist, then the `head` pointer of the booklist will be updated.

Grading Policy:

Part 1 counts for 40% of the overall points in Programming Assignment 2.

Please make sure your program compiles successfully. If not, 10 points will be deducted.

1. **Executability** (5%):
   - Runtime errors: You program must not have runtime errors (e.g., code crash, infinite loop, reading uninitialized memory, accessing the content of a NULL pointer, etc.).
2. **Programming style** (5%):
   - Code efficiency: Code should use the best approach in every case.
   - Readability: Code should be clean, understandable, and well-organized. Please pay special attention to indentation, use of whitespace, variable naming, and organization.
   - Documentation: Code should be well-commented with file header and comments.
3. **Program Specifications/Correctness** (30%):
   Please refer to the Grading Criteria table for details. Specifically, your program should behave correctly, adhere to the instructions, and pass the test program.

file	**item**	**weight (%)**
booklist1.cpp	search_by_id	6
	add_book	6
	remove_book	6
catalog.cpp	fill_catalog	6
proj2p1.docx	Running result	6
	total	**30**