Assignment-06: Arrays Lists and Linked Lists in C

CS 270- Fall 2024

Project Available	***Dec. 02, 2024***	
Component	**Points**	***Due Date: Dec. 11, 2024 (11.59 pm) (Wednesday)***
Linked List Implementation	10	
Array List Implementation	10	
Style and Efficiency	5	
Total	**25**	

Assignment Instructions

The goal of this assignment is to understand and implement the details of a doubly linked list and a list built on top of an array. To achieve these goals, you will complete the implementation of the doubly linked list and the array list, as defined in the .h files provided in the template code. Thoroughly test your implementations using the provided tester programs and extend these tests by adding your own test cases and debugging output where necessary.

1.1 Doubly Linked List

You are required to implement a doubly linked list with sentinel nodes to manage integers. A basic testing program is included in the template zip file. Extend this testing program to verify the correctness of all implemented functions and to handle different edge cases.

- Key Implementation Details:
  - Implement a doubly linked list with sentinel nodes at both the head and the tail.
  - The list must handle integers and provide standard operations, such as:
    - Insertions (e.g., at the head, tail, or at a specified position).
    - Deletions (e.g., removing elements by value or position).
    - Traversal (e.g., iterating through the list from head to tail or tail to head).
    - Search (e.g., finding a node with a specified value).
- Testing:
  - Extend the provided tester program to:
    - Test all implemented functions, including edge cases like inserting or deleting in an empty list.
    - Include debugging output to display the current state of the list after every operation.

1.2 Array as a List

You are required to implement a list-like structure using a dynamic array. This implementation must replicate the behavior of a standard list and dynamically resize the array as needed. A small testing program is included in the template zip file. Extend it to test all operations and edge cases.

- Key Implementation Details:
    - Store all data in a single dynamically allocated array of integers.
    - Use ***realloc*** to resize the array as elements are added or removed.
    - Implement standard list operations, such as:
        - Insertions (e.g., at the end, at a specific index, or in sorted order).
        - Deletions (e.g., by value or index).
        - Search (e.g., finding a value or checking the existence of an element).
        - Traversal (e.g., iterating through the array to display elements).
- Testing:
    - Extend the provided tester program to:
        - Validate all implemented functions and handle edge cases, such as inserting into a full array or deleting from an empty array.
        - Include debugging output to display the array size and contents during each operation.

Submission

Submit the following files in one single zip file via Canvas:

1. Implementation Files:
    - dl_list.c (implementation of the doubly linked list).
    - ar_list.c (implementation of the dynamic array list).
2. Header Files:
    - dl_list.h and ar_list.h.
3. Testing Code:
    - Any changes made to the provided testing/driver program (list_play.c).
    - Any additional test programs you create.
4. Helper Functions:
    - Ensure any helper functions you use are declared in the .c files.

Notes

- Test your code thoroughly for all possible cases to ensure correctness.
- Add sufficient debugging output to verify your code's behavior during testing.
- Use clear and descriptive comments for your code to make it easier to understand.