Doubly-linked List

Goals

- Build a doubly-linked list that utilizes pointers
- Learn how to modify data in a linked list

In this lab, we will create a simple doubly-linked list data structures consisting of Node objects.

Requirements

For simplicity, our doubly-linked list will only contain integer as value.

Note: You are not allowed to use any standard containers for this lab.

Node Class

Node class has the following member variables:

- **next**, a pointer to the next Node object
- **prev**, a pointer to the previous Node object
- **val**, integer value the specific Node contains

Doubly-Linked List

For the doubly-linked list object, it has a **head** pointer that points to the first Node in the linked list, and a **tail** pointer that points to the last Node in the linked list. If the linked list is empty, the head and tail should **point to NULL**. If the linked list has only one Node object, the head and tail should both point to that Node object.

Note: You can decide the class name for doubly-linked list

Functions for the Linked List

Implement the functions to do the following linked list operations:

1. Add a new node to the head.
2. Add a new node to the tail.
3. Delete the first node in the list.
4. Delete the last node in the list.
5. Traverse the list reversely,

that is to print the nodes from back of the linked list to the front (tail to head).

6. Traverse the list,

that is to print the nodes from front of the linked list to the back (head to tail).

Menu

Function #1, #2

When user adds a new node, the program should prompt the user to **enter a number** and **validate the input**, before calling the function.

Operation of adding functions: The function should create a new Node with inputted value stored inside, add the Node object to the list, and **re-assign the head tail pointer** accordingly. After the Node is added, the menu should **call function #6**, which prints the linked list from head to tail.

Function #3, #4

When user chooses to remove a node, the program should **check whether the list is empty**: if so, give a warning message, and if not, proceed with the function.

Operation of removal functions: The function should delete the Node and **free the memory**, then re-assign head/tail pointers as well as the pointers inside adjacent Node of the Node removed.

After the Node is removed, the menu should **call function #6**, which prints the linked list from head to tail.

Function #5

When user chooses to print the list in reverse from tail to head, the program should print the value of all Nodes in the linked list from **tail to head**. **If the list is empty, print a message to indicate that.**

Function #6

Note: Function #6 is special, it does not exist in the menu option, but it gets called every time the user adds or removes a Node from the list.

Whenever this function is called inside the program, it should print the value of all Nodes in the linked list from **head to tail**. **If the list is empty, print a message to indicate that.**

Lastly, you need to add one more option in the menu to **exit the program**.

Important: For this lab, you cannot implement the linked list operations using functions from outside source. You need to write all the functions by yourself!

Extra Credit

Task 1 (10%)

Add two more options:

- Print the value of the Node the head is pointing to
- Print the value of the Node the tail is pointing to

Task 2 (15%)

Create a linked list by reading from a text file:

The program should have an option to create the linked list from a text file you created at the very beginning. And the user can continue to do operations on the newly created linked list.

Create your own text file with some numbers in it. **Make sure** you include the text file in the submission.

The format of the text file is your decision. Some common format is to separate numbers with space characters, or by newline characters.

Note: You can also ask user for the filename they would like to open. This is not required.

Example Run of the Program

Welcome to my linked list!

Choose from following options:

1. Add a new node to the head;
2. Add a new node to the tail;
3. Delete from head;

4. Delete from tail;
5. Traverse the list reversely;
6. Exit.

1

Please enter a positive integer:

15

Your linked list is: 15

Choose from following options:

1. Add a new node to the head;
2. Add a new node to the tail;
3. Delete from head;
4. Delete from tail;
5. Traverse the list reversely;
6. Exit.

1

Please enter a positive integer:

10

Your linked list is: 10 15

Choose from following options:

1. Add a new node to the head;
2. Add a new node to the tail;
3. Delete from head;
4. Delete from tail;
5. Traverse the list reversely;
6. Exit.

2

Please enter a positive integer:

5

Your linked list is: 10 15 5

Choose from following options:

1. Add a new node to the head;
2. Add a new node to the tail;
3. Delete from head;

4. Delete from tail;
5. Traverse the list reversely;
6. Exit.

5

Your linked list is: 5 15 10

Choose from following options:

1. Add a new node to the head;
2. Add a new node to the tail;
3. Delete from head;
4. Delete from tail;
5. Traverse the list reversely;
6. Exit.

4

Your linked list is: 10 15