Programming Assignment 2

Doubly Linked List

In this programming assignment, you will implement a doubly linked list and operations on it. You need to implement functions declared in `main.c`, and perform tasks listed in `main()`. The prototypes of functions are already declared. You should use the declared prototypes. The linked list is supposed to have a dummy node for Head, as we discussed in the class. The below is the screen shot of my implementation when the program is run with random seed 0. Your goal is basically obtaining a similar output with the below.

```
**(TASK 1) Create an empty list
0: NULL
**(TASK 2) insert 5 random numbers at the beginning
1: 38 <-> NULL
2: 7719 <-> 38 <-> NULL
3: 21238 <-> 7719 <-> 38 <-> NULL
4: 2437 <-> 21238 <-> 7719 <-> 38 <-> NULL
5: 8855 <-> 2437 <-> 21238 <-> 7719 <-> 38 <-> NULL
The Element of 5th Node is 38
**(TASK 3) insert a random number at the end
6: 8855 <-> 2437 <-> 21238 <-> 7719 <-> 38 <-> 11797 <-> NULL
**(TASK 4) find the second element
2437 is found
**(TASK 5) insert a random number in the middle
7: 8855 <-> 2437 <-> 8365 <-> 21238 <-> 7719 <-> 38 <-> 11797 <-> NULL
**(TASK 6) delete 2437 from the list
6: 8855 <-> 8365 <-> 21238 <-> 7719 <-> 38 <-> 11797 <-> NULL
**(TASK 7) find 2437 in the list
2437 is not found
**(TASK 8) swap the 2nd and 4th value
6: 8855 <-> 7719 <-> 21238 <-> 8365 <-> 38 <-> 11797 <-> NULL
**(TASK 9) swap the 3rd and 4th value
6: 8855 <-> 7719 <-> 8365 <-> 21238 <-> 38 <-> 11797 <-> NULL
**(TASK 10) reverse the list
6: 11797 <-> 38 <-> 21238 <-> 8365 <-> 7719 <-> 8855 <-> NULL
**(TASK 11) add an element and reverse
7: 11797 <-> 38 <-> 21238 <-> 8365 <-> 7719 <-> 8855 <-> 32285 <-> NULL
7: 32285 <-> 8855 <-> 7719 <-> 8365 <-> 21238 <-> 38 <-> 11797 <-> NULL
**(TASK 12) delete the whole list
0: NULL
**(TASK 13) running time test
The reference time is 2.2580 sec
The elapse time is 5.0640 sec (2.2427 x ref)
```

Please note that the whole program can be run within 10 seconds on my laptop (i7, 16GRAM). If you feel your implementation takes too much time, probably it indicates that your implementation has a problem.

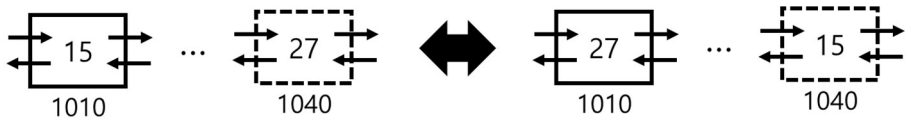
With your implementation, please answer the following question.

1. (TASK 1) Please implement `CreateList()` and `PrintList()`. `CreateList()` returns the dummy head node of the created list without any other node. `PrintList()` show the length of the list and all elements of the list in the order of the list. Please provide the codes of `CreateList()`, `PrintList()`, and the screen shot of TASK 1.

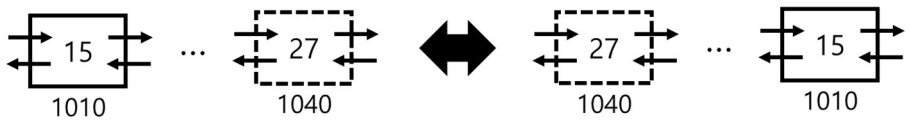
2. (TASK 2~5) Please implement `Insert()`, `Find()`, and `GetNode()`. `Insert(X,L,P)` inserts Element `X` in the list `L` just next to `P`, and returns the inserted node Position, as like `Insert()` of a single linked list in the lecture note. `Find()` is also similarly defined with that of a single linked list in the lecture note. `GetNode(L,idx)` returns the `idx`-th node Position from the first node. Note that `GetNode(L,0)` returns the dummy head node, `GetNode(L,1)` returns the first node, and `GetNode(L,5)` returns the last node if the length of the list is 5. Please provide the codes of `Insert()`, `Find()` and `GetNode()` as well as the screen shot of TASK 2~5.

3. (TASK 6~7) Please implement `Delete()`. `Delete(X,L)` removes the Element `X` from List `L`. If there are multiple nodes with the same `X`, it only removes the first encounter one, just as `Delete()` of single linked list does. How does your implementation is different from that of single linked lists described in the lecture note? Please provide the codes of `Delete()` and the screen shot of TASK 6~7.

4. (TASK 8~9) There can be two ways to swap two nodes. One is swapping elements. In this case, the linked list is not changed. Only the elements are swapped. The other is swapping node objects directly. In this case, each node object holds the same elements, but the locations of the node objects are changed in the list. For these two cases, please implement `SwapElement()` and `SwapNode()`. The former will be simple, but the latter will be somewhat complicated. Give some reasons why swapping nodes is preferred in some cases. Please provide the codes as well as the screen shot of TASK 8~9.



Swap Elements



Swap Nodes

5. (TASK 10~11) Implement `Reverse()` that reverses the list. You may want to use the swap function in the previous problem. Provide the code and the screen shot of TASK 10~11.

6. (TASK 12) Implement `DeleteList()` that deletes all node objects in the list. Provide the code and the screen shot of TASK 12.

7. (TASK 13) Run `perforamce_test()` function to test the actual running time of your implementation. Describe what `perforamce_test()` function does and how the reference time is calculated. Provide the screen shot as well as the measured running time.

8. Set the random seed of `srand()` in `main()` with the last four digits of your student ID. Run the program and provide the whole screen shot.

Please submit **two files**:
(1) **Report**: a single electronic file including answers of the questions, screen shots, source code in the format of doc, hwp or pdf.
(2) **main.c**: a plain-text source code file that can generate the result of the report. Ideally, the screen shot on page 1 can be reproduced with this file.

Note that your assignment will be graded with only the **report**. So, if you do not include source codes in the report, you will not get scores. The plain-text source code will be considered when you have problems, for example, suspicion of the copy&paste or non-sense results. If the plain-text source code is different from codes on the report or it cannot reproduce the results on the report, then your grade will be 0. Please submit the above files through the blackboard system.

If you have any question for this project assignment, please contact me or TA.