EE2331 Data Structure and Algorithm Homework 2

February 17, 2025

Due Time: 11:59PM, Feb. 27, 2025 (Thursday night).
No late work will be accepted.
Submit all files on Canvas using the link for hw2. It should include **three files**: a **typed** txt/pdf/docx file for Problem 1 and two different cpp files for Problem 2 and 3, respectively. Don't zip them.

1. (10 pts) Provide the running time complexity for the following member functions defined in linkedListType.h under folder /Files/notes/note3/. For each member function, write the bigO notation complexity. Again, you should provide the tightest upper bound. For example, if the running time is O(N), you should not write O(N^2). The member functions are: destroyList(), copyList(), initializeList(), operator=, isEmptyList(), print(), length(), search(), insert(), remove(), remove_all(), remove_front(). N is the number of elements in a linked list. If other definitions are needed, you should specify them clearly.

2. (15 pts) One major difference between linked list and array is that an array declares a continuous block of memory. Thus, array allows random access using index (such as array[i]). To examine this difference, implement the following program.

   (a) Read a file testdata.txt, which contains a bunch of int numbers into your program.

   (b) Save these numbers in an array and a linked list, respectively. Use a general linked list, not any special kind of linked list. Don't use any dummy head node. The first node contains the first int, the succeeding node contains the second int, and so on.

   (c) For the array and the linked list, output the address of each element. For example, suppose your array has name A. Output the **address** of A[0], A[1], .. A[n-1] for n-size array. Suppose your

linked list has head pointer head. Output every node's address. Observe whether the array elements' addresses are continuous. Observe whether the nodes' addresses are continuous. **Write a short summary about your observation in your program**. The output format should look like:
"The address of array element A[0] is ..."
"The address of array element A[1] is ..."
...
"The address of array element A[n-1] is ..."
"The addresses of the nodes in the linked list are: ...".

(d) Write your summary in a comment section such as:
/*
I found that the array elements addresses .....
*/

Grading: Note that we will test your program using different files with the same input format, but not the same content. Correct implementation and summary: 15 points. The summary part: 5/15 points. If your program does not work correctly but contains roughly correct logic, 3 pts.

3. (20 pts) Write a function that takes a sorted linked list as input and rearranges it such that all **even-indexed nodes appear before the odd-indexed nodes**. The relative order of odd-indexed nodes and even-indexed nodes should be maintained. For example, given the list (1, 3, 5, 6, 10), the function should modify the list to become (3, 6, 1, 5, 10). Note that the first element has index 1.

   Input: a sorted linked list that will be created by you after reading the input file (see below).

   Output: the re-arranged linked list. In your program, you must print out the linked list after conducting the re-arrangement.

   In order to create a sorted linked list, you can first read a file that contains sorted list. Two sample files are provided at Canvas/homework/hw2: test1.txt and test2.txt. We can change the contents of the test file when testing your program. Ensure that you test your program using different input files with the same format. In addition, a file reading cpp program is provided to give you a quick start about the first step.

   Grading: Correct implementation: 20 points. If your program does not work correctly but contains roughly correct logic, 5 pts.