**Linked Lists in Action**

In this homework you will implement a permutation generator using Linked Lists in C++. Specifically, you will generate a single permutation of the first N integers, e.g., \([2, 4, 5, 1, 3, 0]\) for \(N = 6\).

First create your template class for the Linked List data structure as described in the lecture slides. We will test this class with various data types; so implementing a template class is mandatory.

[30 points] To fill the next list node, generate random numbers between 0 and \(N-1\), until you get one number that is not already in the list. Print the resulting list \(L_1\) to screen.

[30 points] A more efficient way to generate permutations is to randomly swap the elements of an initial list. To keep things more interesting, you will swap nodes at odd indices (assuming the first node is at index 0 which is even). Specifically, you will swap index 1 and 3, then 5 and 7, and so on. Make sure you swap the nodes by updating the pointers, not just swap the data values in the nodes. Initialize with \(L_1\) from the first part. Print \(L_1\) and then print the resulting swapped version \(L_2\).

[30 points] In this version start with \(L_2\), and remove all the nodes at even indices. Create a new list \(L_3\) which is filled by these removed nodes. Merge \(L_2\) and \(L_3\) into \(L_4\) by taking one node from \(L_2\) and two from \(L_3\) until you finish \(L_2\). Print \(L_2\), \(L_3\), and \(L_4\).

[10 points] Return the index of the max element in a list, assuming the first element sits at index 0.

In addition to the typical Linked List functions (insert, remove, etc.), you will have to implement a swap() and a getMax() function (plus helper functions, if needed) in your Linked List class.

Submission: Email to ys@ceng.metu.edu.tr your source code files along with a readme.txt that reports the execution time of each part for \(N = 10, 10^2,\) and \(10^3\). Comment on the time complexity of the parts.