Assignment 4

CS-AD-103: Data Structures

Assignments are to be submitted in groups of two or three. Upload the solutions on NYU classes with one PDF file for the theoretical assignments and separate C++ files (just source code) for each coding assignment.

Problem 1 (10 points)**.**
In the previous assignment, you wrote a program that takes an arithmetic expression in infix notation and evaluates it. You used a `Stack` class in your program that was implemented using an array. This imposed the restriction that the maximum size of the stack needed to be fixed when an object of `Stack` type was created.

In this assignment, you need to change the implementation of the `Stack` class and use linked lists instead of arrays. The only public members of the class should be corresponding to the operations that a stack allows (push, pop, top, isEmpty). In particular, all the details related to the linked list should be private to the class. Do not forget to provide suitable constructors and a destructor.

Check that your program still evaluates an expression input in infix notation correctly.

Problem 2 (10 points)**.**
Write a C++ program with the following functions:

- A function `createList` that inputs a sequence of integers from the users until the number 0 is entered, and puts the numbers in a singly linked list *in the order in which the numbers arrive*. The function should take no arguments and return a pointer to the first element in the list (the **start** pointer).

- A function `PrintList` that takes a pointer to the first element (the **start** pointer) in a linked list and prints the numbers in the list in the order in which they appear in the list.

- A function `Reverse` that takes a reference to the **start** pointer of a linked list and reverses the linked list. The function should update the start pointer appropriately. The function should use the same nodes as in the original linked list i.e., you should not create any new nodes.

Test these functions by creating a list, reversing it and then printing it in your `main` function. You can use the following definition of a Node class:

```
class Node {
public:
  int data;
  Node *next;
};
```

Problem 3 (10 points)**.** Write a C++ function whose declaration is "`Node* mergeSort(Node *start);`", where the class `Node` is as in the previous exercise. The function takes a pointer to the first node in a linked list, sorts the linked list using the merge sort algorithm and returns a pointer to the first node in the sorted list. Your algorithm should run in time $O(n \log n)$ where n is number of nodes in the input linked list.

Problem 4 (10 points)**.** Suppose that you are given the **start** pointer to a linked list that some other programmer has created. As such a linked list is supposed to end with `nullptr`, but you suspect that due to programming errors, the linked list may contain a cycle as shown in the figure. Give an algorithm (pseudocode suffices) to check if this is the case. The algorithm should take $O(n)$ time where n is the number of distinct nodes in the list and it should use only a *constant amount of memory* (apart from the storage used for the list).

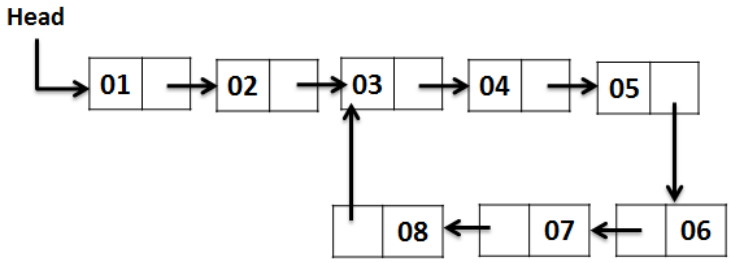

Figure 1: Linked list with a cycle.