Programming Assignment - 4
CS-200 Dr. Mian Muhammad Awais
Total Marks: 100.
Due date: 12th December 2020 11:55 PM

All instructions are the same as previous assignments. Please go over them if you haven't read them. You need to submit two .cpp files in a zipped folder. Plagiarism/cheating rules apply.

Naming convention:
For questions:
<rollNumber>_Q1.cpp
<rollNumber>_Q2.cpp

Zipped Folder:
<rollNumber>_<AssignedTAName>_PA4.zip

Question 1 [50]:
Given a (singly) linked list with head node root, write a function to split the linked list into k consecutive linked list "parts".
The length of each part should be as equal as possible: no two parts should have a size differing by more than 1. This may lead to some parts being null.
The parts should be in the order of occurrence in the input list, and parts occurring earlier should always have a size greater than or equal parts occurring later.
Return a List of List Nodes representing the linked list parts that are formed.

Examples

1->2->3->4, k = 5 // 5 equal parts [[1], [2], [3], [4], null]

Input:
root = [1, 2, 3], k = 5
Output: [[1],[2],[3],[],[]]
Explanation:
The input and each element of the output are ListNodes, not arrays.
For example, the input root has root.val = 1, root.next.val = 2, \root.next.next.val = 3, and root.next.next.next = null.
The first element output[0] has output[0].val = 1, output[0].next = null.
The last element output[4] is null, but it's string representation as a ListNode is [].

Input:
root = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], k = 3
Output: [[1, 2, 3, 4], [5, 6, 7], [8, 9, 10]]
Explanation:
The input has been split into consecutive parts with size difference at most 1, and earlier parts are a larger size than the later parts.

The function prototype will be something like splitListToParts(ListNode* root, int k), however, it’s not necessary to use this.
Marks distribution:
Code works on all inputs: 50 marks
Code works on some but not all inputs: 25 marks
Works on no input: 0 marks

Question 2 [50]:
In this question, you will implement a music playlist using a recursive singly linked list. You need to edit the pa4.cpp file provided in the assignment for this question, don't forget to rename it according to the naming convention before submitting.

The node will be a struct called 'Node' which has 3 variables:

1. Song_title
2. Id
3. Pointer to Node next

Your playlist will be a linked list of these nodes.
You will need to write functions to:

1. Insert a song at the end (recursive definition) [5]
2. Insert a song after a specific song (iterative definition) [5]
3. Delete a song (from the end) (recursive definition) [5]
4. Display the playlist (recursive definition) [5]
5. Find the min id of the songs in the playlist (recursive definition) [10]
6. Determine the length of the playlist (recursive definition)[5]
7. Search for a song in the playlist (search by song title, search by song id) (recursive definition) [5]
8. Delete the playlist (recursive definition)[10]

You must implement the functions iteratively or recursively as mentioned above and in the driver code. You are not allowed to make changes to the function definitions.

Make the appropriate prompts to the user for entering the details of the songs where required and also to determine whether the user wants to search for a song by its title or its id.

In the main function, you will create a menu that will allow the user to avail of any of these functionalities and will continue doing so until the user desires to stop.

Ensure that your program makes the appropriate checks where required so that your program never crashes unexpectedly.