21-805-0206: Lab 3 - Data Structures Lab

Assignment 3

Instructions:

1. All programs have to be submitted as .cpp files that can be compiled and executed without error. The input & output shall be included in the same file as comments
2. A separate pdf file named as "**rollno_firstName**" shall be submitted for questions that require an explanation (marked with *). This should be **hand written.**
3. Compress all the above as a single .zip file and upload in the moodle page before **25/08/23**

Part 4: Linked List

1. Create a singly linked list of size n that contains positive integers. Write functions to implement the following operations in the list:

   search (key) – Search for the occurrence of the element specified by key in the list and return its position. Display all positions if key is present multiple times. Print NOT FOUND if key is not present.
   count_duplicates (key) - Search for the occurrence of the element specified by key in the list. Display the number of times key is duplicated. Print NO DUPLICATES if key is present only once.
   remove_duplicates (key) - Search for the occurrence of the element specified by key in the list. Keep only the first occurrence of key and delete every other duplicate entry.
   insert_pos (key, pos) – Insert the element specified by key at position pos in the list and return the new list. pos is a postive integer. Display 'LIST TOO SMALL' if list has less than pos - 1 entries initially.

   Input format:
   The first line of the input is the number of elements in the list n.
   The next line contains n space separated integers constituting the list.
   Each of the next line of the input contains a character from {'s', 'c', 'r', 'i', 'e'} followed by one or two non-negative integers.
   - '*s key*' means search for the occurrence of the element specified by key in the list.
   - '*c key*' means display the number of times key is duplicated in the list.
   - '*r key*' means delete all duplicate entries of key in the list keeping only the first occurrence of key.
   - '*i key pos*' means insert the element specified by key to position pos in the list.
   - 'e' means exit the program.

Input	Output
6 10 1 0 14 5 45 s 11 i 5 7 s 5 i 50 9 i 5 2 c 5 r 5 e	 NOT FOUND 10 1 0 14 5 45 5 5 7 LIST TOO SMALL 10 5 1 0 14 5 45 5 2 (and not 3) 10 5 1 0 14 45

2. Write a function to implement *alternate splicing* of a singly linked list. Given a linked list consisting of integers, split it into two smaller sublists, such that each of the sublists contain alternating elements in the original list. The elements in the newly created sublists must be in the same order as they occur in the original list.

   Example:
   Original list is {6, 4, 10, 13, 1, 7, 88, 10, 5}
   Newly created sublists are {6, 10, 1, 88, 5} and {4, 13, 7, 10}

   Input is a single line with n space seperated integers and output should be in two lines , each line coresponding to a sublist. Please note the first line should be the sublist with first element.

3. Write a program to convert a given singly linked list to a doubly linked list, by adding the required back pointers. You may add an additional ***tail*** pointer to point to the end of the list. Traverse and print the list contents both from ***head*** to ***tail***, as well as ***tail*** to ***head*** .