Shri G. S. Institute of Technology and Science Department
Of Computer Engineering
CO_____: DATA STRUCTURES

Lab Assignment # 02 (Linked List)

Submission Date: 30 Dec 2019@23:59

Late Submission: Not allowed
No copying allowed. If found then students involved in copying will fail this course.

1. Write ADT of Singular Linked List (SLL) program with insertNode, deleteNode and printList functions.
2. Write ADT of Doubly Linked List (DLL) program with insertNode, deleteNode and printList functions.
3. Given a singly linked list of characters, write a function *int isLinkedListPalindrome(Node * head)* that returns 1 if the given list is a palindrome, else 0.
4. Write a function *Node * mergeLinkesLists(Node * head1, Node * head2)* that takes two sorted linked lists and merges them into one linked list, then returns the head of new linked list.
5. Write a *void Count(Node* head, int number)* function that counts the number of times a given int occurs in a list.
6. Design a data structure for the *unbounded integers*. This data structure also support the different arithmetic operations (add, subtraction, multiplication etc.) on unbounded integers. Using unbounded integer data structures, write the following program:

    a. Find the factorial of the very large number.
    b. Find the value of x^x, using unbounded integers.

7. Write a program for the Indore Election problem of **Assignment-1** using linked list.
8. Perform the following operations on the linked list:

    c. Perform pair-wise swapping of nodes of a given list.

Input:
20 -> 18 -> 15 -> 10 -> 8 -> 6 -> 5 -> 3 -> 7 -> NULL
Output:
18 -> 20 -> 10 -> 15 -> 6 -> 8 -> 3 -> 5 -> 7 -> NULL

    d. Remove alternate nodes from a given linked list.

Input:
20 -> 18 -> 15 -> 10 -> 8 -> 6 -> 5 -> 3 -> NULL
Output:
20 -> 15 -> 8 -> 5 -> NULL

e. Given a pair linked lists, insert nodes of second linked list into the first linked list at alternate positions. Assume that the first linked list has at least as many elements as the second.

Input:
1 -> 2 -> 3 -> NULL
4 -> 5 -> NULL
Output:
1 -> 4-> 2 -> 5 -> 3 -> NULL

9. Write a program to represent a polynomial in variable X using a singly linked list, each node of which contains two kinds of data fields; one to store coefficient and another stores the power on variable X. For example, if a polynomial is: P(X) = $2X^2$
+ 3X + 4 then the structure of linked list is as below:

| 2 | 2 | -> | 3 | 1 | -> | 4 | 0 |
Perform polynomial addition on two such polynomials represented by linked lists and generate a new list representing the sum of those polynomials.

Input:
| 2 | 2 | **->** | 3 | 1 | **->** | 4 | 0 | **->** NULL
| 1 | 3 | **->** | 2 | 1 | **->** | 1 | 0 | **->** NULL
Output:
| 1 | 3 | **->** | 2 | 2 | **->** | 5 | 1 | **->** | 5 | 0 | **->** NULL

10. Consider a set of friends are on facebook. Each of them have a uniqu ID and their own list of friends.

Let's consider a structure for each linked list node and facebook user as bellow struct

linked_list

```
{
int data;
struct linked_list *next;
};
typedef struct linked_list NODE; struct
        fbUser{
   int userID; NODE* friends;
   struct fbUser *next;
};

typedef struct fbUser FBU;
```

Write functions to find the following and our input will be a list of struct fbUser which has their uniqe ID and their corresponding friends list.

1. Linked_list of strangers for user 'x' at level 1

- We consider friends of friends as friends. i.e. if x is friend with y and z, then friends of y and z are also friends with x.
- Only 1 level of friends are considered as friends. i.e. if x is friend with y, and y is friend of z, then only z is friend with x not the friends of z.

2. Linked_list of strangers for user 'x' at level 2

- We consider friends of friends of friend as friends. i.e. if x is friend with y and z, then friends of y and z are also friends with x.
- Only 2 level of friends are considered as friends. i.e. if x is friend with y, and y is friend of z, and z is friend of w then only w is friend with x not the friends of w.

Consider the following example. #id Name

Friends_id

0 Avik {1,2}
1 Barath (2,3,4,5,6)
2 Chetty {4,5,6,7,8)
3 Dinash {6,7,8,9,10,11,12,13,14}
4 Elik {12,13,14,15,16,17,18,19,20 }
…….
9 Ira {101,102,103}

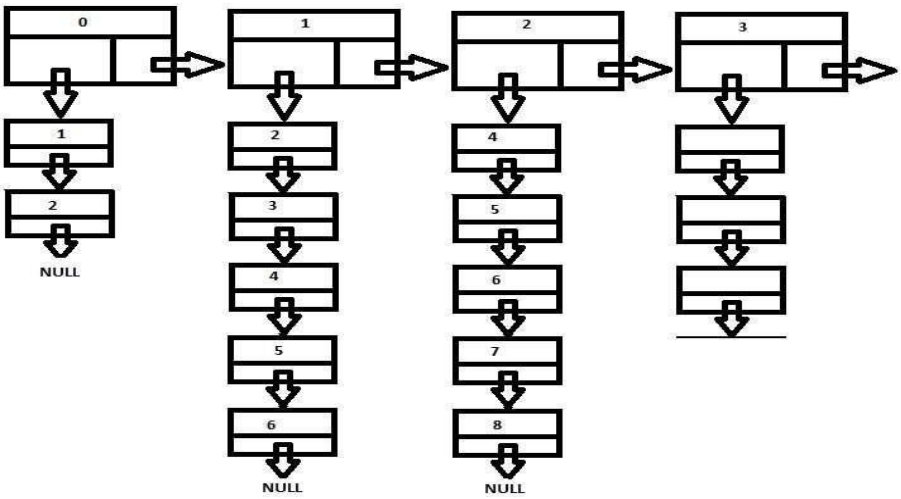
1. Strangers-1 to Avik(1)

- Direct friends {1,2}
- Friends of friends (1 level) {3,4,5,6,7,8}

- So, not strangers-1 {1,2,3,4,5,6,7,8}
- Strangers-1 are { 9,10,11,12,13,14}

2. Strangers-2 to Avik(1)

- Direct friends {1,2}
- Friends of friends (1 level) {3,4,5,6,7,8}
- Friend of friends of friend(2 level) {9,10,11,12,13,14,15,16,17,18,19,20 }
- So, not strangers-2 {1,2,3,4,5,6,7,8, 9,10,11,12,13,14,15,16,17,18,19,20 }
- Strangers-2 are {101,102,103}




```
NODE* find_strangers_1(int person_id, FBU *list_of_fbusers)
{
 //Please write your code here
}
NODE* find_strangers_2(int person_id, FBU *list_of_fbusers){
 //Please write your code here
      }
```