Assignment 2: Linked lists - 1

1. For a given singly linked list insert a node:
    a. at the beginning
    b. at the end
    c. at a given position k

    Input:

    ```
    value=8, k=4
    1 -> 2 -> 5 -> 7 -> 4 -> NULL
    ```

    Output:

    ```
    8 -> 1 -> 2 -> 5 -> 7 -> 4 -> NULL
    1 -> 2 -> 5 -> 7 -> 4 -> 8 -> NULL
    1 -> 2 -> 5 -> 8 -> 7 -> 4 -> NULL
    ```

2. For a given singly linked list delete a node:
    a. at the beginning
    b. at the end
    c. at a given position k

    Input:

    ```
    k = 3
    1 -> 2 -> 5 -> 7 -> 4 -> NULL
    ```

    Output:

    ```
    2 -> 5 -> 7 -> 4 -> NULL
    1 -> 2 -> 5 -> 7 -> NULL
    1 -> 2 -> 7 -> 4 -> NULL
    ```

3. Implement insertion sort using singly linked list.

4. Reverse a singly linked list.

    Input:

    ```
    1 -> 2 -> 5 -> 7 -> 4 -> NULL
    ```

    Output:

    ```
    4 -> 7 -> 5 -> 2 -> 1 -> NULL
    ```

5. Write a program for printing the following in a given linked list:
    a. maximum
    b. minimum
    c. maximum – minimum

6. Print the sum of all even numbers stored in a circular linked list.

7. Take N numbers as input from the user and create a doubly linked list.

8. Find the smallest number in the doubly linked list.

9. Delete the smallest number in the doubly linked list.

10. Reverse a doubly linked list.

11. Print all the elements at the index of multiples of k with the first element assumed to have an index of 0. Do this for a single pass of the linked list.

    Input:
    ```
    k=3
    12 -> 15 -> 18 -> 17 -> 19 -> 20 -> 22 -> NULL
    ```
    Output:
    ```
    12 -> 17 -> 22 -> NULL
    ```

12. Extend the above solution assuming that the list is circular and the N-th index is the same as 0-th index. You may need multiple passes. However, every number should be printed only once during its first selection.

    Input:
    ```
    k=3
    12 -> 15 -> 18 -> 17 -> 19 -> 20 -> 22 -> NULL
    ```
    Output:
    ```
    12 -> 17 -> 22 -> 18 -> 20 -> 15 -> 19 -> NULL
    ```

13. Delete duplicate elements from a given linked list. Retain the earliest entries.

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

14. Reverse a linked list in groups of given size.

    Input:
    ```
    k=4
    1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 12 -> 7 -> NULL
    ```
    Output:
    ```
    4 -> 3 -> 2 -> 1 -> 8 -> 7 -> 6 -> 5 -> 7 -> 12 -> NULL
    ```

15. Remove in a linked list all the nodes that have a greater value to their right.

    Input:
    ```
    10 -> 12 -> 15 -> 20 -> 5 -> 16 -> 25 -> 8 -> NULL
    10 -> 12 -> 15 -> 20 -> 25 -> 26 -> 30 -> 40 -> NULL
    20 -> 18 -> 15 -> 10 -> 8 -> 6 -> 5 -> 3 -> NULL
    ```
    Output:
    ```
    20 -> 25 -> 8 -> NULL
    40 -> NULL
    20 -> 18 -> 15 -> 10 -> 8 -> 6 -> 5 -> 3 -> NULL
    ```

16. Remove alternate nodes from a given linked list.

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

17. Perform pair-wise swapping of nodes of a given linked list.

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

18. Find the middle element of a given linked list. In case of tie print the second one.

    Input:

    ```
    5 -> 7 -> NULL
    5 -> 7 -> 17 -> NULL
    5 -> 7 -> 17 -> 13 -> NULL
    5 -> 7 -> 17 -> 13 -> 11 -> NULL
    ```

    Output:

    ```
    7
    7
    17
    17
    ```

19. Check whether a given singly linked list is palindrome or not.

    Input:

    ```
    a -> b -> NULL
    a -> b -> a -> NULL
    s -> a -> g -> a -> r -> NULL
    r -> a -> d -> a -> r -> NULL
    ```

    Output:

    ```
    not palindrome
    palindrome
    not palindrome
    palindrome
    ```

20. Calculate the frequency of occurrence of each element in a given linked list in the same order they appear. Avoid printing multiple entries.

    Input:

    ```
    20 -> 18 -> 15 -> 20 -> 6 -> 18 -> 5 -> 18 -> NULL
    ```

    Output:

    ```
    Freq(20) = 2
    Freq(18) = 3
    Freq(15) = 1
    Freq(6) = 1
    Freq(5) = 1
    ```

21. Create two circular linked lists and find their maximum numbers. Merge the two circular linked lists such that the maximum number of 2nd circular linked list immediately follows the maximum number of the 1st circular linked list.

    Input:

    ```
    12 -> 28 -> 18 -> 25 -> 19 -> NULL
    5 -> 24 -> 12 -> 6 -> 15 -> NULL
    ```

    Output:

    ```
    12 -> 28 -> 24 -> 12 -> 6 -> 15 -> 5 -> 18 -> 25 -> 19 -> NULL
    ```

22. 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
    ```

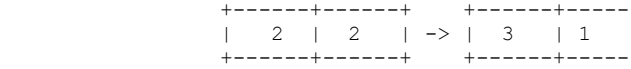
23. Swap k-th node from the beginning with k-th node from the end in a linked list.

    Input:
    ```
    k=1   1 -> 2 -> 3 -> NULL
    k=2   1 -> 2 -> 3 -> NULL
    k=3   1 -> 2 -> 3 -> NULL
    ```
    Output:
    ```
    3 -> 2 -> 1 -> NULL
    1 -> 2 -> 3 -> NULL
    3 -> 2 -> 1 -> NULL
    ```

24. Make a function that adds a linked list to itself at the end.

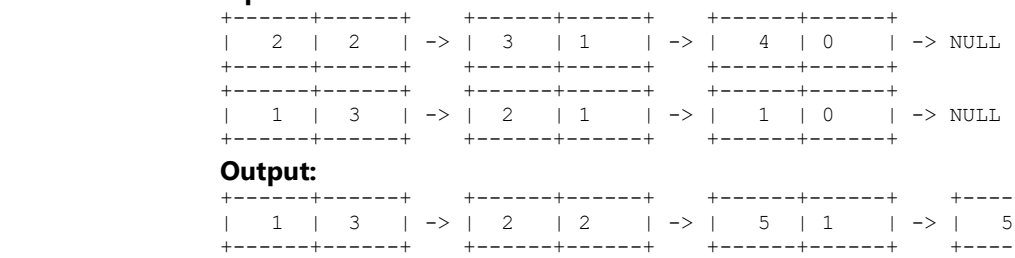
    Input:
    ```
    4 -> 2 -> 1 -> NULL
    ```
    Output:
    ```
    4 -> 2 -> 1 -> 4 -> 2 -> 1 -> NULL
    ```

25. 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:

    Perform polynomial addition on two such polynomials represented by linked lists and generate a new list representing the sum of those polynomials.

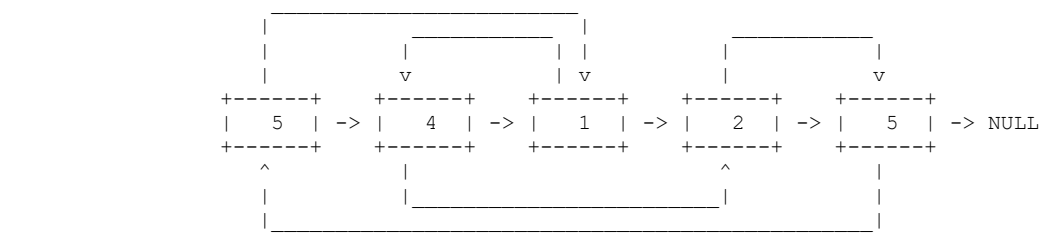
    Input:

    Output:

26. Create a linked list in which each node has two pointers: FRIEND and NEXT. FRIEND can point to any other node in the linked list. Make a duplicate linked list with the same values and connections.

    Input: (original linked list)
    ```
    5 -> 1; 1 -> 4; 4 -> 2; 2 -> 5; 5 -> 5
    ```

    Output: (new linked list)
    ```
    5 -> 1; 1 -> 4; 4 -> 2; 2 -> 5; 5 -> 5
    ```

27. Create a linked list using array of structures. The next pointer is replaced now by index of the next element. Implement `malloc_arr()` and `free_arr()` functions to give a new or free an element. So, the structure will have `data` and `index` instead of `data` and `next` pointer.