CL-2001
Data Structures
Lab # 03

Objectives:

1. Link List
    a. Insertion
    b. Deletion
    c. Finding
    d. Traversal
2. ADT

Note: Carefully read the following instructions (*Each instruction contains a weightage***)**

1. There must be a block of comments at start of every question's code; the block should contain brief description about functionality of code
2. Proper indentation of code is essential
3. Variable name should be meaningful
4. Make a Microsoft Word file and past all of your C++ code with screenshot of outputs in MS word.
5. First think about statement problems and then write/draw your logic on copy.
6. After copy pencil work, code the problem statement on MS Studio C++ compiler.
7. At the end when you done your today lab tasks, attached only MS word file and make your submission on Google Classroom.
8. Late and email submission is not accepted. All tasks must be submitted during the lab time.

Problem 1:

Write a function **rearrange(list)** that takes a ListADT object as input and rearranges its elements in such a way that all even elements appear before all odd elements. The relative order of even and odd elements should be maintained. The function should return the rearranged ListADT object. Write the rearrange(list) function using the provided ListADT operations and ensure that it works correctly for various data types. You are given a ListADT class that supports the following operations: **get**(), **insert**(), **remove**(), **removeAt**(), **replace**(), **size**(), **isEmpty**(), and **isFull**().

Problem 2:

Write two separate functions to find the intersection and union of two ListADT objects. The intersection function should return elements common to both lists, while the union function should return a list containing all unique elements

Problem 3: ***(Concatenating Lists)***

Write a program that concatenates two linked list objects of integer. The program should include function concatenate, which takes references to both list objects as arguments and concatenates the second list to the first list.

Problem 4: ***(Merging Ordered Lists)***

Write a program that merges three unordered list objects of integers into a single ordered list object of integers. Function merge should receive references to each of the list objects to be merged and reference to a list object into which the merged elements will be placed.

Problem 5: ***(Summing, Averaging, Maximum and Minimum Elements in a List)***

Write a program that generates a linked list with 30 random integers between 0 and 1000 (inclusive). The program should then calculate and display the following statistics:

1. The sum of all elements in the linked list.
2. The average of the elements in the linked list.
3. The maximum value among the elements in the linked list.
4. The minimum value among the elements in the linked list

Problem 6: ***(Copying a List in Reverse Order)***

Write a program that creates a linked list object of 15 characters and creates a second list object containing a copy of the first list, but in reverse order.

Problem 7: ***(Inserting Anywhere)***

Write a program that creates an odd number of nodes in a linked list. Then, find the middle of the linked list and insert a data item into the list. After that, if the number of nodes becomes even, repeat the same process to insert a new node.

Problem 6:

Write a RemoveDuplicates() function that takes a list sorted in decreasing order and removes any duplicate nodes from the list. The goal is to traverse the list only once. Additionally, count the number of duplicates found and display it at the end of the program.

Problem 7:

Write a SortedInsert() function which given a list that is sorted in decreasing order, and a single node, inserts the node into the correct sorted position in the list. While Insert() allocates a new node to add to the list, SortedInsert() takes an existing node, and just rearranges pointers to insert it into the list.