Data Structures

Fall 2021

Lab 05

Learning outcomes:

In this lab you are expected to learn the following:

- Node based Linked list
- Recursion
- Google testing

Task 1:

Design a class linked List with data members Data and next pointer.

Your class must provide implementation for the following

- A default constructor for Node
- A parameterized constructor
- Getters for data and next
- Setters for data and next

Design a linkedList class with a pointer of type Node. This will carry the address of **head** initially, later it will be required for traversal.

Your class must provide implementation for the following

- A default constructor
- **Insert** function to insert data item. For every new data element to be inserted, you will have to reach up to the end point of LL and then perform insertion
- **Insert to head** function to insert data element at the start of your LL
- Design **isEmpty** to check if there is no data element in the LL
- Why don't you need to design isFull?
- A **Search** function to search for any element provided by user
- **Update** function to update/replace one data element with another
- A function **Insert at index** that will count data elements in the LL and then place a new data element in the specified location. To know exactly after which node we have to make insertion, we need count the elements and that will help us to identify index (index doesn't exist actually, but we are using this term to get to know the location for insertion)
- A function **Delete** to delete the specified data element, for this you will have to save the address of the next node that appears after the one to be deleted. Saving the address will help you after deleting the element to concatenate the list with the elements appearing after the element that is supposed to be deleted.
- A function to **print** the data of LL

Task 2:

Reuse the Task 1 and create 2 linkedlists L1 and L2, with data elements mentioned below.

L1= {1, 3, 5, 7} //contains odd numbers
L2= {2, 4, 6, 8} //contains even numbers
Write a function **mergeLists** to merge the elements of L2 with L1 in such a way that the resultant list L1 should like the one below:

L1= {1, 2, 3, 4, 5, 6, 7,8}

Task 3:

After completing the basic operations over LL, write a recursive function **isPalindrom** that checks if a given list is a palindrome or not.