

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI (RAJ.)**  
**CS F111 Computer Programming**  
**LAB SESSION #12**  
(Linked Lists)

**Q1.** The exercise is to create a linked list that stores floating-point values and implement all the operations as follows:

- **createNewList**: initializes a new linked list and returns it.
- **createNewNode**: creates a new linked list node for an element passed to it and returns the node.
- **insertNodeAtStart**: inserts a node at the beginning of the list
- **insertNodeAtEnd**: inserts a node at the end of the list
- **insertNodeAfterElem**: inserts a node after a specific element in the list
- **removeNodeAtStart**: removes first node of the list
- **removeNodeAtEnd**: removes last node of the list
- **removeElement**: removes the node containing an element passed as argument
- **printList**: prints all the elements contained in the list
- **find**: finds whether a given element exists in the list or not

Given a zip file - linkedlist\_code.zip. You can find function declarations and structure definitions in **linkedList.h**. You will have to implement the operations in **linkedList.c**. The **main.c** file contains the main() function which calls the functions defined in **linkedList.c**. You can follow the way the functions are called in the main() function to implement them in **linkedList.c**. You are free to modify anything in all three files. Please complete implementing the functions in **linkedList.c**. Please refer to slides to understand the logic and code for implementing linked lists.

**Q2.** Instead of storing float values in the linked list, you should now store student records containing the attributes: **ID** (char array), **Name** (char array), **Dept** (char array), **math\_marks** (integer), **phy\_marks** (integer), **chem\_marks** (integer). Then implement the following functions with specifications as follows:

- **createNewList**: initializes a new linked list for storing student records
- **createNewNode**: creates a new linked list node that can store a student record taken as parameter
- **insertNodeAtStart**: inserts a node containing a student record at the beginning of the list
- **insertNodeAtEnd**: inserts a node containing a student record at the end of the list
- **insertNodeAfterElem**: inserts a node after a specific element in the list. That specific element can be identified with the **ID** of the student record. This function must take that **student ID** as an input parameter along with a new node to insert.
- **removeNodeAtStart**: removes first node of the list
- **removeNodeAtEnd**: removes last node of the list
- **removeElement**: removes the node containing a student record identified by an ID. That **ID** must be taken as an input parameter, along with other parameters.
- **printList**: prints all the elements contained in the list. Note that each element is a student record. So create a separate function **printStudentRecord** that takes a student record and prints it. **printList** can simply call **printStudentRecord** to print the students records stored in the linked list.
- **find**: finds whether a student with given **ID** exists in the linked list or not

Create multiple header (.h) and source (.c) files to keep the program modular. Have all the global variables and function declarations in **linkedList\_Student.h**. The implementations of all the functions can be in **linkedList\_Student.c**. The main function can be in **main\_StudentList.c** where you can create a linked list, append nodes to it, remove nodes from it using appropriate function calls that are defined in *linkedList\_Student.c*. Also give a script file to compile, link and execute the project.