DATA STRUCTURES LAB	
Course Code: ISL36	**Credits: 0:0:1**
Pre – requisites: Fundamentals of Computing	**Contact Hours: 14 P**
Course Coordinator: Dr. Kusuma S	

Course Contents

Unit 1

List of Experiments:

1. Write a C program to store and manage a list of records student records (student name, usn , department and total marks) using structures. Using dynamic memory allocation and pointers write functions to create student records, add the records, display and search records based on usn.
2. Write a C program to implement a Stack data structure using arrays, where each pushed number represents a web page visited. When a new web page is visited, push its page number onto the stack. When the user moves backward, pop and display the page number removed from the stack. Use a structure to represent the Stack. Handle and display appropriate messages for stack overflow and underflow conditions.
3. Write a C program to convert and print a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and binary operators + - * /. Apply the concept of stack data structure to solve this problem.
4. Write a C program to evaluate a valid postfix expression using stack. Assume that the postfix expression is read as a single line consisting of non-negative single digit operands and binary operators. The operators are + - * and /.
5. Write a C program to simulate a call center phone system using a queue data structure as an array. The system should store incoming customer calls in the order they arrive and provide service based on this order (FIFO). The program must include options to add a new phone call to the queue and remove a call when it is being serviced. Display appropriate messages for queue overflow and underflow situations.
6. Write a C program to simulate process scheduling using a circular queue implemented with arrays. The program should allow the user to add a process, remove a process, and display all processes in the circular queue. Ensure that appropriate messages are shown for queue overflow and underflow conditions.
7. Write a C program to implement a singly linked list to store customer IDs in a store billing system. Each time a customer arrives for billing, insert the customer ID at the end of the list. Once the billing is completed, remove the customer ID from the beginning of the list. Demonstrate these operations and display the updated linked list after each action.
8. Write a C program to read a polynomial expression from the user in sorted order of exponents. Use a singly linked list and implement an “insert at end” function to store each term of the polynomial. Write a function to add two polynomials and return the resulting polynomial to the main program. Display all the input polynomials as well as the final added polynomial.
9. Write a C program to simulate file memory allocation in an operating system using a doubly linked list. Each node of the list should represent a file number. Implement the following operations: a. Insert a new file at the front of the doubly linked list (representing a file being newly created and placed in memory). b. Delete a file from a specified position in the list. c. Display the list of files along with their allocated memory sizes.

10. Write a C program to simulate forward and backward web browsing using a Stack implemented with a linked list. Each node in the stack should represent a web page visited. Implement stack operations to handle the following actions: Back (pop the current page and move to the previous page), Forward (push pages when navigating forward), Visit a new page (push a new node onto the stack). Demonstrate all operations and display the updated list of visited pages after each action. Ensure appropriate messages are shown in case of stack underflow.
11. Write a C program to implement a Circular Queue data structure using a circular linked list for a simple real-world application, such as managing customers waiting in a service counter. The program should support enqueue and dequeue operations and display the updated queue after each action. Ensure that appropriate messages are shown for queue underflow or other exceptions.
12. Write a C program to simulate an image-indexing system for a hospital database using a Binary Search Tree (BST). Each CT image is represented by a unique numeric ID. Insert these image IDs into the BST to build an efficient indexing structure. Implement preorder traversal to display the indexed image IDs, showing how the system retrieves stored images. Additionally, include a feature to identify and display the largest image IDs in the BST, representing the highest indexed images in the database.