CSS 2101 DATA STRUCTURES

TUTORIALS

Tutorial-1

1. Using pointers, traverse a static array and count how many elements are even and how many are odd.*(Use case*: Processing sensor values or survey data.)

2. Write a function to reverse the elements of a static array **in-place** using pointers.(*Use case*: Undoing operations or reversing user-entered data.)

3. Given marks of 5 students in 3 subjects (use 2D array), use pointers to:

- Calculate total marks of each student
- Find average marks of each subject

(*Use case*: Academic report generation)(Note: Not to use structures)

Tutorial-2

1. Solve Tower of Hanoi for n disks using Recursion and print the moves.(Use Case: Recursive problem-solving patterns, puzzle-solving algorithms.)
2. Dynamically allocates memory using calloc() for an array of integers to store attendance status of n students (where n is entered by the user).
   - 1 → Present
   - 0 → Absent
   - Initially, all values should be set to zero (absent).
   - Then, accept a list of roll numbers of students who are present and update their status in the array.
   - Finally, display the list of present and absent students based on their roll numbers.
3. Given a dynamically allocated 2D array, write functions to:
   - Print a specific row.
   - Print a specific column.
   - Replace all elements of a selected row with a given value.

   (*Use Case*: Spreadsheet row/column operations.)

Tutorial - 3

Define a structure for student records with:

- Roll Number (int), Name (string), Marks in 3 Subjects (array of 3 integers),Total Marks (int calculated),Average Marks (float — calculated)
  - Use Arrays of structures (for student lists)
  - Pointers to structures (for traversal and manipulation)
  - Recursive functions (for searching and sorting operations)

1. Write a recursive Bubble Sort function to sort student records in ascending order of Roll Numbers. (Use pointer arithmetic to swap structures.)
2. After sorting the student list by Roll Number, write a recursive function to search for a student by Roll Number using Binary Search.
3. Implement Recursive Insertion Sort to sort students by their Average Marks (descending).
4. Write a recursive function that counts how many students passed in all subjects (assuming pass mark = 40).

Tutorial 4

Q1. A college wants to store details of participants in a coding competition. Each participant record should contain:

- Participant ID (integer)
- Name (string)
- Score (integer)
- **Task:**
- Write a program to **create the Singly linked list recursively** by taking user input until the user chooses to stop.
- Write a **function** to traverse the linked list and display participant details in the ascending order with respect to id.

Tutorial 4

Q2. A library maintains two separate lists of newly arrived books — one list for Fiction (List X1) and another for Non-Fiction (List X2). Each book record contains:

- Book ID (integer)
- Title (string)
- Author (string)

Write a program to concatenate the two doubly linked lists so that X1 points to the first node of the combined list. After concatenation, display the full list of books in descending order with respective to id.

Tutorial 5

A train has coaches connected in both directions (forward and backward), which can be represented as a **Circular Doubly Linked List (CDLL)**.

Write a menu-driven program to manage the train by performing the following operations:

Insertion

- Add a new coach at the **front** of the train.
- Add a new coach at the **end** of the train.
- Add a new coach at a **specific position** in the middle of the train.

Deletion

- Remove the **first coach** from the train.
- Remove the **last coach** from the train.
- Remove a coach at a **specific position** in the middle of the train.

Display

- Display all coaches in **forward direction** (engine to rear).
- Display all coaches in **reverse direction** (rear to engine).

Tutorial 6

Multiply Two Polynomials Using Circular Doubly Linked List (with Header Node)

i. Represent each polynomial using a circular doubly linked list with a header node.

ii. Multiply each term of the first polynomial with every term of the second polynomial.

iii. Merge like terms during or after multiplication.

Tutorial 7

Q1. Convert the following infix expression into its equivalent postfix expression using the stack method.
While solving, show only the **stack contents at each step** (do not write a program). **(K+L- M*N+(O^P)*W/U/V *T+Q)**

Q2. Convert the following infix expression into its equivalent prefix expression using the stack method.
While solving, show only the **stack contents at each step** (do not write a program). **A+B-C*D+(E^F)*G/H/I*J+K**