**Day 14 docs**

**Task - 1:**

**What do you understand by traversing a linked list?**

Traversing a linked list means visiting each node in the list, one by one, starting from the head (first node), until you reach the end (null). This is typically done to:

Display the values of the nodes

Search for a specific value

Perform operations like summing elements, counting nodes, etc.

Starts with the head node.

Prints the data of the current node.

Moves to the next node (current = current.next).

Repeats this until it reaches the end of the list (null).

**Task 09:**

**What are the methods of the stack class.. List them down.. With a one liner..**

| **Method** | **One liner Description** |
| --- | --- |
| push(E item) | Adds (pushes) an item to the top of the stack. |
| pop() | Removes and returns the top item of the stack. |
| peek() | Returns the top item without removing it. |
| isEmpty() | Returns true if the stack contains no elements. |
| search(Object o) | Returns the 1-based position of an item from the top (or -1 if not found). |
| size() | Returns the number of elements in the stack. |
| clear() | Removes all elements from the stack. |
| contains(Object o) | Returns true if the stack contains the specified element. |
| iterator() | Returns an iterator to traverse the stack elements. |
| get(int index) | Gets the element at the specified index (since Stack extends Vector). |

Home task:

**Task - 10:**

**What is the difference between recursion and iteration.**

| Feature | Recursion | Iteration |
| --- | --- | --- |
| Definition | A function that calls itself | Repeating a block of code using loops |
| Code Style | More compact and elegant | Often longer but more straightforward |
| Used When | Problem can be broken into smaller subproblems | Repetitive tasks with a known number of steps |
| Speed | Generally slower (due to function call overhead) | Generally faster |
| Memory Use | Uses call stack → more memory | Uses fewer resources, often just variables |
| Example | Factorial via f(n) = n \* f(n-1) | Factorial via for (i = 1 to n) |
| Risk | May cause StackOverflowError if too deep | Safer for large input sizes |
| Best For | Tree, Graph, Divide & Conquer, Backtracking | Simple loops and counting problems |