**LAB SESSION 2: STACK DATA STRUCTURE**

**AIM**: To implement expression converter and evaluator using Stack Data Structure.

**PROBLEM DEFINITION:**

1. Develop a C program to implement the following:
   * Convert infix to postfix
   * Convert infix to prefix
   * Evaluate a postfix expression
   * Evaluate a prefix expression

**THEORY:** A stack is a data structure that follows the LIFO (last in first out) principle. In this data structure we have a TOP pointer which refers to the element at top of the stack. To retrieve or delete data from the stack we have only one position that is the TOP position.



Fig. 2.1: Basic Working of Stack Data Structure

There are some basic operations that allow us to perform different actions on a stack.

* Push: Insert an element to the top of a stack
* Pop: Delete an element from the top of a stack
* IsEmpty: Check if the stack is empty
* IsFull: Check if the stack is full
* Peek: Get the value of the top element without removing it

**Working of Stack Data Structure**

The operations work as follows:

* A pointer/location called TOP is used to keep track of the top element in the stack.
* When initializing the stack, we set TOP value to -1 so that we can check if the stack is empty by comparing TOP == -1 (for array implementation or TOP = NULL for LL implementation)
* On pushing an element, we increase the value of TOP and place the new element in the position pointed to by TOP.
* On popping an element, we return the element pointed to by TOP and reduce its value.
* Before pushing, we check if the stack is already full
* Before popping, we check if the stack is already empty

**LAB SESSION 3: QUEUE DATA STRUCTURE**

**AIM**: To implement a task scheduler using Queue Data Structure.

**PROBLEM DEFINITION:** Create a structure TASK with the following data fields:

TaskId(int)

TaskTitle (char [])

TaskDuration (int) to be interpreted in seconds

Status: Idle/Queued/Completed

Create an array of TASK comprising of 10 tasks. (Preferably Take input from file or initialise at compile time)

Implement a Queue data structure to schedule tasks from the above array. Max size of Queue is 5. Your options should include the following:

1. Enter the task id of the tasks to be scheduled (insert these in a queue). If status is queued or completed then do not allow scheduling. If the queue is full, ask the user to wait for a certain amount of time (Print the minimum time of waiting and the max waiting time to the user)

2. Run the Tasks: delete the task at start of queue and run it for the given taskDuration. (Use delay() function to suspend the running of the program for the indicated time). Once the task is completed prompt the user the menu and continue based on his choice.

3. Display the details of tasks that are queued up.

**THEORY:** A queue is a data structure that follows the FIFO (first in first out) principle. In this data structure we have a Head and a tail pointer which refers to the element at start and end of the queue. To retrieve or delete data from the stack we have only one position that is the TOP position.

A queue is an object (an abstract data structure - ADT) that allows the following operations:

* Enqueue: Add an element to the end of the queue
* Dequeue: Remove an element from the front of the queue
* IsEmpty: Check if the queue is empty
* IsFull: Check if the queue is full
* Peek: Get the value of the front of the queue without removing it

