## While folks are joining

Get you laptops ready and login to www.crio.do.

We will be coding away in the session!

# DSA-1

## Session 8

## What’s for this session?

- Stack and Queue
  - Introduction
  - Library methods
- Problems
  - Evaluating a Postfix Expression
  - Perform Queue Operations

## What are Collections?

- Collections are libraries with implementations of DS with commonly needed methods

- You don’t need to implement the Data Structures from scratch while solving problems. Use available Collections.

## Stack

- **Last in First Out**
  - **Stack of Plates,** Books stacked, Deck of cards
  - Only put new plate of top, remove from top
- **Properties**
  - Values can repeat
  - Represented using Array or Linked List
  - Insert/Delete/Lookup only at top
  - Search - will have to remove all and insert back
  - Sort - not possible
- **When to use**
  - Behavior required is stack behavior

## Stack

- Applications
  - Stacks can be used for **expression evaluation**.
  - Stacks can be used to **check parenthesis matching** in an expression.
  - Stacks can be used for Conversion from one form of expression to another.
  - Stacks can be used for Memory Management. **Function Call Stack. Recursion.**
  - Stack data structures are used in **backtracking** problems.
- Library methods In different languages
  - Refer to resources here
  - empty() - It returns true if nothing is on the top of the stack. Else, returns false.
  - peek() - Returns the element on the top of the stack, but does not remove it.
  - pop() - Removes and returns the top element of the stack.
  - push(element) - Pushes an element on the top of the stack.

Note: Stack will be covered in more detail in a pack of its own, later on

## Stack - Frequently asked problems

- How to identify Stack problems?
  - If you need to keep track of the previous elements in the order in which they occurred.

- Frequently asked problems
  - Tree traversals without using recursion
  - Postfix, Infix and Prefix conversions
  - Expression evaluation
  - Balanced Parentheses problem
  - Next Greater Element

## How to Approach Problems?

For any given problem, following these milestones will help you solve the problem systematically:

- **Milestone 1** - Understand the problem statement and confirm your understanding with some examples or test cases, including edge cases.
- **Milestone 2** - Think about approaches and select the best one you know. Explain your approach to a 10 year old. Write the pseudocode with function breakdown.
- **Milestone 3** - Expand pseudocode to code
- **Milestone 4** - Demonstrate that the solution works

## Activity 1 - Evaluating a Postfix expression

## Queue

- **First In First Out**
  - A **Queue of people** waiting to be served
  - Can only join at the back and leave at the front
- **Properties**
  - Values can repeat
  - Represented using Array or Linked List
  - Insert only at the end
  - Delete only at the beginning
  - Lookup only at the beginning
  - Search is expensive
- **When to use**
  - Behavior required is Queue behavior
- **Types**
  - Simple Queue, Circular Queue, Priority Queue (SORT), Doubly Ended Queue

## Queue

- Applications
    - Queues - Kafka, RabbitMQ
    - The consumer who comes first to a shop will be served first.
    - CPU task scheduling and disk scheduling.
    - Waiting list of tickets in case of bus and train tickets.
- Library methods In different languages
    - Refer to resources here
    - add() - Inserts the specified element into the queue.
    - peek() - Returns the head of the queue.
    - remove() - Returns and removes the head of the queue.
    - poll() - Returns and removes the head of the queue.

Note: Hash will be covered in more detail in a pack of its own, later on

## Queue - Frequently asked problems

- How to identify Queue problems?
  - Queue problems usually call for use of Queue explicitly

- Frequently asked problems
  - Level Order Traversal without using recursion
  - BFS without using recursion
  - Queue Implementation using a Linked List
  - Implement a stack using the queue data structure
  - Implement a queue using the stack data structure

## Activity 2 - Perform Queue Operations

## Questions?

Take home exercises

- Check balanced parentheses sequence

To be solved before the next session on Saturday, 11 AM

## Feedback

Thank you for joining in today. We’d love to hear your thoughts and feedback.

https://bit.ly/dsa-nps

# Thank you