Lu Phone Maw 6511157

Lut Lat Aung 6511163

Thu Ya Hlaing 6511238

**Stack Data Structure and Algorithms**

**Abstract**

This document provides an overview of the fundamental concepts of the "Stack" data structure and its associated algorithms. A stack is a linear data structure that follows the Last-In-First-Out (LIFO) principle, where the last element added to the stack is the first one to be removed. This document explores the basic operations of a stack, its applications, and common algorithms used with stacks.

**1. Stack Data Structure**

**1.1 Definition**

A stack is a linear data structure that consists of a collection of elements with two primary operations: push and pop. The stack follows the Last-In-First-Out (LIFO) order, meaning that the most recently added item is the first one to be removed.

**1.2 Operations**

The two fundamental operations of a stack are:

* **Push:** This operation adds an element to the top of the stack.
* **Pop:** This operation removes and returns the element from the top of the stack.

Additionally, stacks typically support the following operations:

* **Peek (or Top):** This operation returns the element at the top of the stack without removing it.
* **isEmpty:** This operation checks whether the stack is empty or not.
* **Size:** This operation returns the number of elements currently in the stack.

**1.3 Implementation**

A stack can be implemented using various data structures, such as arrays or linked lists. The choice of implementation depends on the specific use case and the requirements.

**2. Applications of Stacks**

Stacks are versatile data structures with numerous applications in computer science and software engineering, including:

**2.1 Expression Evaluation**

Stacks are commonly used to evaluate arithmetic expressions, particularly infix expressions. By converting infix expressions to postfix notation, you can efficiently evaluate them using a stack.

**2.2 Function Call Stack**

In programming languages, a stack is used to manage function calls and their associated data. Each function call creates a new stack frame, and the stack keeps track of the call hierarchy.

**2.3 Undo/Redo Functionality**

Stacks are used to implement undo and redo functionality in applications. Each action is pushed onto the undo stack, allowing users to revert to previous states.

**2.4 Backtracking Algorithms**

Backtracking algorithms, such as depth-first search (DFS), often use stacks to maintain a history of visited nodes or states.

**2.5 Memory Management**

Operating systems use stacks for managing function call stacks, as well as for handling memory allocation and deallocation.

**3. Common Stack Algorithms**

**3.1 Reverse a String**

One of the simplest stack algorithms is reversing a string. You can push each character onto the stack and then pop them off to obtain the reversed string.

**3.2 Valid Parentheses**

Stacks are used to check the validity of expressions with parentheses. You can push opening parentheses onto the stack and pop them when a closing parenthesis is encountered, ensuring proper nesting.

**3.3 Infix to Postfix Conversion**

Converting infix expressions to postfix notation using a stack is essential for efficient expression evaluation.

**3.4 Depth-First Search (DFS)**

DFS, used in graph traversal, employs a stack to keep track of visited nodes and explore deeper into a graph before backtracking.

**3.5 Tower of Hanoi**

The Tower of Hanoi puzzle can be solved using recursion and a stack to manage the intermediate states.