QUAID-E-AWAM UNIVERSITY OF ENGINEERING, SCIENCE AND TECHNOLOGY

# DEPARTMENT OF ARTIFICIAL INTELLIGENCE

Dated: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Roll No(s): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Lab 03 – Arrays and Stacks

### Lab Objectives

• Understand how arrays and lists work in Python.

• Implement basic array operations such as reversing, rotating, and merging.

• Learn how to implement and use stacks in Python using both lists and linked lists.

• Solve problems like balanced parentheses checking and postfix expression evaluation using stacks.

### Background

Arrays are a fundamental data structure used to store elements of the same type. In Python, lists serve as dynamic arrays that can grow or shrink in size. Mastering list manipulation is essential before moving to more advanced structures.

A stack is a linear data structure that follows the LIFO (Last In, First Out) principle. It is widely used in recursion, parsing, and expression evaluation.

### Lab Tasks

#### Task 1 – Dynamic Arrays

# Dynamic Array Example  
arr = []  
n = int(input("Enter number of elements: "))  
for i in range(n):  
 val = int(input(f"Enter element {i+1}: "))  
 arr.append(val)  
  
print("Array elements:", arr)

#### Task 2 – Reverse, Rotate, and Merge Arrays

# Reverse array  
arr = [1, 2, 3, 4, 5]  
print("Reversed array:", arr[::-1])  
  
# Rotate array  
k = 2  
rotated = arr[k:] + arr[:k]  
print("Rotated array:", rotated)  
  
# Merge arrays  
a = [1, 3, 5]  
b = [2, 4, 6]  
merged = a + b  
print("Merged array:", merged)

#### Task 3 – Implement Stack Using List

stack = []  
  
def push(element):  
 stack.append(element)  
 print(f"Pushed: {element}")  
  
def pop():  
 if not stack:  
 print("Stack Underflow")  
 else:  
 print(f"Popped: {stack.pop()}")  
  
def peek():  
 if stack:  
 print(f"Top Element: {stack[-1]}")  
  
push(10)  
push(20)  
push(30)  
peek()  
pop()  
peek()

#### Task 4 – Implement Stack Using Linked List

class Node:  
 def \_\_init\_\_(self, data):  
 self.data = data  
 self.next = None  
  
class Stack:  
 def \_\_init\_\_(self):  
 self.top = None  
  
 def push(self, data):  
 new\_node = Node(data)  
 new\_node.next = self.top  
 self.top = new\_node  
 print(f"Pushed: {data}")  
  
 def pop(self):  
 if self.top is None:  
 print("Stack Underflow")  
 else:  
 print(f"Popped: {self.top.data}")  
 self.top = self.top.next  
  
s = Stack()  
s.push(5)  
s.push(10)  
s.pop()

#### Task 5 – Balanced Parentheses Checker

def is\_balanced(expr):  
 stack = []  
 pairs = {')':'(', '}':'{', ']':'['}  
 for ch in expr:  
 if ch in '({[':  
 stack.append(ch)  
 elif ch in ')}]':  
 if not stack or stack.pop() != pairs[ch]:  
 return False  
 return not stack  
  
expr = "{[()]}"  
print("Balanced" if is\_balanced(expr) else "Not Balanced")

#### Task 6 – Postfix Expression Evaluation

def evaluate\_postfix(expression):  
 stack = []  
 for ch in expression.split():  
 if ch.isdigit():  
 stack.append(int(ch))  
 else:  
 b = stack.pop()  
 a = stack.pop()  
 if ch == '+': stack.append(a + b)  
 elif ch == '-': stack.append(a - b)  
 elif ch == '\*': stack.append(a \* b)  
 elif ch == '/': stack.append(a / b)  
 return stack[0]  
  
expr = "5 3 + 8 2 - \*"  
print("Postfix Result:", evaluate\_postfix(expr))

### Questions

1. What is the main difference between a list and an array in Python?

2. Why is a stack called a LIFO structure?

3. Modify Task 5 to print the position of the first unmatched parenthesis.

4. What would be the output of the postfix expression 4 5 \* 8 2 / + ?

5. Implement a 'Peek' operation in the linked list-based stack.