#### Stack Implementation Using Linked List (stack\_linkedlist\_implementation.py)

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

def pop(self):

if self.is\_empty():

print("Stack underflow!")

return None

popped = self.top

self.top = self.top.next

popped.next = None

return popped.data

def peek(self):

if self.is\_empty():

print("Stack is empty!")

return None

return self.top.data

def is\_empty(self):

return self.top is None

def display(self):

current = self.top

while current:

print(current.data, end=" -> ")

current = current.next

print("None")

#### Stack Implementation Using Array (stack\_array\_implementation.py)

class Stack:

def \_\_init\_\_(self, capacity):

self.capacity = capacity

self.stack = []

def push(self, data):

if len(self.stack) < self.capacity:

self.stack.append(data)

else:

print("Stack overflow!")

def pop(self):

if self.is\_empty():

print("Stack underflow!")

return None

return self.stack.pop()

def peek(self):

if self.is\_empty():

print("Stack is empty!")

return None

return self.stack[-1]

def is\_empty(self):

return len(self.stack) == 0

def display(self):

print("Stack elements:", self.stack)

#### Problem 1 for Stack (stack\_problem1.py): Implementing Stack Operations

from stack\_linkedlist\_implementation import Stack

if \_\_name\_\_ == "\_\_main\_\_":

stack = Stack()

stack.push(10)

stack.push(20)

stack.push(30)

print("Stack after pushes:")

stack.display()

popped\_element = stack.pop()

print(f"Popped element: {popped\_element}")

print("Stack after pop:")

stack.display()

top\_element = stack.peek()

print(f"Top element: {top\_element}")

#### Example Problem 2: Balanced Parentheses Using Stack (stack\_problem2.py)

def is\_balanced(expression):

stack = Stack()

for char in expression:

if char == '(':

stack.push(char)

elif char == ')':

if stack.is\_empty():

return False

stack.pop()

return stack.is\_empty()

if \_\_name\_\_ == "\_\_main\_\_":

expression = "(())"

if is\_balanced(expression):

print("The parentheses are balanced.")

else:

print("The parentheses are not balanced.")