QUIZE 2

Question 1: Evaluate the following postfix expression using stack data structure.

Postfix Expression:

2 + ((5 + 4) * 3) + 1

Let's break it down step by step. We'll use a stack to evaluate the expression.

Push 2 onto the stack:
Stack = [2]
Push 5 onto the stack:
Stack = [2, 5]
Push 4 onto the stack:
Stack = [2, 5, 4]
Encounter "+" (addition):
Pop 4 and 5 from the stack, add them:
5 + 4 = 9
Push the result (9) back onto the stack:
Stack = [2, 9]
Push 3 onto the stack:
Stack = [2, 9, 3]
Encounter "*" (multiplication):
Pop 3 and 9 from the stack, multiply them:
9 * 3 = 27
Push the result (27) back onto the stack:
Stack = [2, 27]
Encounter "+" (addition):
Pop 27 and 2 from the stack, add them:
2 + 27 = 29
Push the result (29) back onto the stack:
Stack = [29]
Push 1 onto the stack:
Stack = [29, 1]
Encounter "+" (addition):
Pop 1 and 29 from the stack, add them:
29 + 1 = 30
Push the result (30) back onto the stack:
Stack = [30]

Answer: The result is 30.

Question 2: Consider a stack A. What will be in the stack A after the following operations?

Operations:

Push(5):
Stack = [5]
Push(10):
Stack = [5, 10]
Pop():
Pop the top element (10):
Stack = [5]
Push(2):
Stack = [5, 2]
Push(5):
Stack = [5, 2, 5]

Push(Pop() + Pop()):

Pop 5 and 2, then add them:

5 + 2 = 7

Push the result (7) back onto the stack:

Stack = [5, 7]

Pop():

Pop the top element (7):

Stack = [5]

Push(9):

Stack = [5, 9]

Push(1):

Stack = [5, 9, 1]

Push(Pop()):

Pop 1 and push it back onto the stack:

Stack = [5, 9, 1]

Answer: The final stack is [5, 9, 1].

Question 3: Using stack, check if the following expression is correct or not.

Expression:

a + (b + 3) - ((c * d) + e

To check if this expression has balanced parentheses, we use a stack.

a: No parentheses, ignore.

+: No parentheses, ignore.

(: Push onto stack: Stack = ['(']

b: No parentheses, ignore.

+: No parentheses, ignore.

3: No parentheses, ignore.

): Pop from stack to match the opening parenthesis: Stack = []

-: No parentheses, ignore.

(: Push onto stack: Stack = ['(']

(: Push onto stack: Stack = ['(', '(']

c: No parentheses, ignore.

*: No parentheses, ignore.

d: No parentheses, ignore.

): Pop from stack to match the opening parenthesis: Stack = ['(']

+: No parentheses, ignore.

e: No parentheses, ignore.

At the end, there is still one unmatched opening parenthesis in the stack, which means the expression is not balanced.

Answer: The expression is incorrect.

Question 4: Convert the following infix expression to postfix expression using stack.

a. Infix: 8 * 2 - 3

8: Operand, add directly to the postfix expression: Postfix = 8

*: Operator, push onto the stack: Stack = [*]

2: Operand, add directly to the postfix expression: Postfix = 8 2

-: Operator, pop from the stack until encountering an operator with lower precedence (pop * and add to the postfix expression):

Postfix = 8 2 *

Then push - onto the stack: Stack = [-]

3: Operand, add directly to the postfix expression: Postfix = 8 2 * 3

Pop - from the stack and add to the postfix expression: Postfix = 8 2 * 3 -

Converted Postfix Expression: 8 2 * 3 -

b. Infix: 1 * 3 * 3 + 5 + 1

1: Operand, add directly to the postfix expression: Postfix = 1

*: Operator, push onto the stack: Stack = [*]

3: Operand, add directly to the postfix expression: Postfix = 1 3

*: Operator, pop from the stack (pop * and add to the postfix expression):

Postfix = 1 3 *

Then push * onto the stack: Stack = [*]

3: Operand, add directly to the postfix expression: Postfix = 1 3 * 3

+: Operator, pop from the stack (pop * and add to the postfix expression):

Postfix = 1 3 * 3 *

Then push + onto the stack: Stack = [+]

5: Operand, add directly to the postfix expression: Postfix = 1 3 * 3 * 5

+: Operator, pop from the stack (pop + and add to the postfix expression):

Postfix = 1 3 * 3 * 5 +

Then push + onto the stack: Stack = [+]
1: Operand, add directly to the postfix expression: Postfix = 1 3 * 3 * 5 + 1
Pop + from the stack and add to the postfix expression: Postfix = 1 3 * 3 * 5 + 1 +

Converted Postfix Expression: 1 3 * 3 * 5 + 1 +