University of Djelfa

Computer Science Department **Time 1h30.**

Algorithms and Data Structures Exam (L2)

NAME……………………………….:

EXERCISE 01 (06 MARKS)

Implement a function to insert a new node with a given value at position k in a singly linked list (0-based indexing). The function should handle the following cases:

- Insert at the head (k = 0).
- Insert in the middle.
- Insert at the end.
- Insert in an empty list.

If k is out of bounds, raise an appropriate error message.

SOLUTION:

EXERCISE 02 (04 MARKS)

Reverse a doubly linked list.

SOLUTION:

Choose one problem and solve it:

PROBLEM 01 (10 MARKS)

Write a program that evaluates a regular infix arithmetic expression (with operators +, -, *, /, and parentheses) by first converting it to postfix notation and then evaluating it using a stack.

Steps:

1. Convert the given infix expression to postfix (Reverse Polish Notation).
2. Evaluate the resulting postfix expression using a stack.

Input:

- A string representing a valid infix arithmetic expression. The expression will contain:
    - **Operands**: Integers.
    - **Operators**: +, -, *, /.
    - **Parentheses** for grouping.

Output:

- The result of the evaluated expression as an integer.

Constraints:

- The input expression is valid and contains no spaces.
- The operators +, -, *, / have the usual precedence, with parentheses overriding the precedence.

PROBLEM 02 (04 MARKS)

Implement a stack data structure using two approaches:

1. **Single Queue Approach**: Use only one queue to implement the stack functionality, ensuring that the most recently added element is always at the front of the queue.
2. **Two Queues Approach**: Use two queues to implement the stack. One queue will hold the stack elements, and the second will serve as a temporary helper during the push operation.

For both approaches, implement the following stack operations:

- push(x): Adds the element x to the stack.
- pop(): Removes and returns the top element of the stack.
- peek(): Returns the top element without removing it.
- empty(): Checks if the stack is empty.

Tasks:

1. Write the implementation for both approaches.
2. Compare the approaches by:
    - Counting the number of iterations (loops or re-queued elements) needed for push operations.
    - Listing the number of variables used in each approach.

SOLUTION:	

💯