자료구조 과제 3

20231609 정희선

Problem 1.

1. **push function**
    - Inputs: Stack array, pointer to top index, item to push
    - Action: Adds an item to the stack.
2. **pop function**
    - Inputs: Stack array, pointer to top index
    - Output: Item popped from the stack
    - Action: Removes and returns the top item from the stack.
3. **getToken function**
    - Inputs: Pointer to symbol, pointer to index `n`, input string
    - Output: Token type based on the symbol
    - Action: Reads a character from the input string and classifies it as an operator, operand, or end-of-string.
4. **printToken function**
    - Input: Token type
    - Output: Corresponding character for the token
    - Action: Converts a token type back to its corresponding character.
5. **postfix function: Infix expression -> Postfix expression**
    - Input: Infix expression `input`
    - Output: Postfix expression `postfix_exp`
    - Initialize `stack`
    - Iterate through the input expression:
        - For operands: Directly append to `postfix_exp`
        - For operators: Compare priorities using `stack` and manage accordingly
        - For parentheses: Manage using `stack` to process operators inside
    - Append remaining operators in `stack` to `postfix_exp`

- Return the completed `postfix_exp`

6. **eval function: Evaluate Postfix expression**

    - Input: Postfix expression `postfix_exp`
    - Output: Computed result
    - Initialize `stack`
    - Iterate through the Postfix expression:
        - For operands: Push onto `stack`
        - For operators: Pop necessary operands from `stack`, perform operation, then push result back onto `stack`
    - Pop the final result from `stack` and return

7. **main function**

    - Receive Infix expression from user input
    - Call `postfix` function to generate Postfix expression
    - Call `eval` function to evaluate the Postfix expression
    - Print results

Problem 2

1. **push function**
2. **pop function**
3. **getToken function**
4. **printToken function**

1-4 의 function 은 problem 1 과 동일하다.

5. **prefix function**

    - Inputs: Infix expression string, output array for prefix expression
    - Action: Converts an infix expression to a prefix expression using a stack for operators and handling parentheses correctly.

6. **main function**

- Action: Prompts user for infix expression, calls `prefix` to convert it, and prints the resulting prefix expression.

Problem 3

1. **init_stk function**
    - Input: Pointer to `Stack`
    - Initializes the `Stack` by setting the top to `NULL`.
2. **push function**
    - Input: Pointer to `Stack`, integer `data`
    - Adds a new `Node` with `data` to the top of the stack.
3. **pop function**
    - Input: Pointer to `Stack`
    - Output: Data of the top node
    - Removes the top node from the stack and returns its data.
4. **is_empty function**
    - Input: Pointer to `Stack`
    - Output: Boolean indicating if the stack is empty
5. **print_stk function**
    - Input: Pointer to `Stack`
    - Converts the stack data to an integer by popping elements and multiplying by increasing powers of ten, then prints the result.
6. **delete_num function**
    - Input: Character array `num`, integer `k`
    - Uses a `Stack` to generate the smallest possible number by removing `k` digits.
    - Iterates over each digit in `num`, using the stack to maintain the smallest possible number by comparing the top of the stack with the current digit and deciding whether to pop or keep the stack's top.
7. **main function**

	• Manages user input and calls `delete_num` to execute the functionality of the program.