**LAB REPORT ON**

**Data Structures**



**Lab No. 2**

**Date: 2020/09/11**

**Topic: Stack**

**Submitted By**  **Submitted To**

Tapendra Pandey Bibha Staphit

BCT-D Department of Electronics

075BCT093 and Computer Engineering

**ALGORITHMS**

I.Push and pop in stack

1: Declare stack abc

2: Push to stack

3: While abc is not empty

Get top element

4: Pop from stack

II. Convert infix to postfix expression

1. Scan the infix expression from left to right.

2. If the scanned character is an operand, output it.

3. Else,

…..3.1 If the precedence of the scanned operator is greater than the precedence of the operator in the stack(or the stack is empty or the stack contains a ‘(‘ ), push it.

…..3.2 Else, Pop all the operators from the stack which are greater than or equal to in precedence than that of the scanned operator. After doing that Push the scanned operator to the stack. (If you encounter parenthesis while popping then stop there and push the scanned operator in the stack.)

4. If the scanned character is an ‘(‘, push it to the stack.

5. If the scanned character is an ‘)’, pop the stack and and output it until a ‘(‘ is encountered, and discard both the parenthesis.

6. Repeat steps 2-6 until infix expression is scanned.

7. Print the output

8. Pop and output from the stack until it is not empty.

III. Convert infix operation to prefix operation

1: Reverse the infix expression , ‘(‘ = ‘)’ and ‘)’ =‘(‘.

2: Obtain the postfix expression of the modified expression

3: Reverse the postfix expression

IV:Evaluate postfix operation

1:Create a stack to store operands (or values).

2:Scan the given expression and do following for every scanned element.

a) If the element is a number, push it into the stack

b) If the element is a operator, pop operands for the operator from stack. Evaluate the operator and push the result back to the stack

3: When the expression is ended, the number in the stack is the final answer

V: Check for balanced paranthesis

1:Declare a character stack S.

2:Now traverse the expression string exp.

If the current character is a starting bracket (‘(‘ or ‘{‘ or ‘[‘) then push it to stack.

If the current character is a closing bracket (‘)’ or ‘}’ or ‘]’) then pop from stack and if the popped character is the matching starting bracket then fine else parenthesis are not balanced.

3:After complete traversal, if there is some starting bracket left in stack then “not balanced”