Institute of Computer Technology

B. Tech. Computer Science and Engineering

Sub: DS (2CSE302)

Practical-5

Objectives: To learn applications of stack using **infix** to **postfix** conversion and **postfix** expression evaluation.

1. Rohan is a 7th semester, who is studying at GUNI-ICT. During his *"Compiler Design"* course, his course faculty explained him that compiler work differently while it does evaluation of an expression due to below reasons:

- Infix expressions are readable and solvable by humans because of easily distinguishable order of operators, but compiler doesn't have integrated order of operators.
- Hence to solve the Infix Expression compiler will scan the expression multiple times to solve the sub-expressions in expressions orderly which is very in-efficient.
- To avoid this traversing, Infix expressions are converted to postfix expression before evaluation.

a) Write the c program to convert below infix expression into postfix using stack.
   i. a-b*c
   ii. (a-b)*c+(d+f)

Hint:

- Infix expression can be represented with C+D, the operator is in the middle of the expression.
- In postfix expression, the operator will be at end of the expression, such as CD+
- Use isalnum() function, which checks whether the given character is alphanumeric or not. **isalnum**() function defined in **ctype.h** header file.
- Alphanumeric: A character that is either a letter or a number
- Postfix expression conversion
  - **Input**: a-b*c , **Output**: a b c * -
  - **Input**: (a-b)*c+(d+f), **Output**: a b - c * d f + +

b) Rohan understood that why the conversion of the infix expression to postfix expression is important. Then, his friend Shyam asked him to evaluate the below postfix expression using stack using c program.
   i. 237+*
   ii. 53-8*13+/

Hint:

- Postfix expression evaluation
    - **Input**: 237+*, **Output**: 20
    - **Input**: 53-8*13+/, **Output**: 4