**Experiment No 9:**

**A program to convert infix expression to postfix expression**

**Aim:** A program for converting an infix expression to postfix using STACK.

**Theory:**

The way to write arithmetic expression is known as a notation. An arithmetic expression can be written in three different but equivalent notations, i.e., without changing the essence or output of an expression. These notations are −Infix NotationPrefix (Polish) NotationPostfix (Reverse-Polish) NotationThese notations are named as how they use operator in expression.  
**Infix Notation**We write expression in infix notation, e.g. a - b + c, where operators are used in-between operands. It is easy for us humans to read, write, and speak in infix notation but the same does not go well with computing devices. An algorithm to process infix notation could be difficult and costly in terms of time and space consumption.**Prefix Notation**In this notation, operator is prefixed to operands, i.e. operator is written ahead of operands. For example, +ab. This is equivalent to its infix notation a + b. Prefix notation is also known as Polish Notation.**Postfix Notation**This notation style is known as Reversed Polish Notation. In this notation style, the operator is post fixed to the operands i.e., the operator is written after the operands. For example, ab+. This is equivalent to its infix notation a + b.

**Expression evaluation:**Stack data structure is used to evaluate expressions in infix, postfix, and prefix notations. Operators and operands are pushed onto the stack, and operations are performed based on the stack’s top elements.

**\*Any Solved Example.**

**Algorithm:**

Suppose Q is an arithmetic expression written in infix notation. This algorithm finds the equivalent postfix expression P.

1.Push “(“on to STACK, and add “)” to the end of Q.2.Scan Q from left to right and repeat Steps 3 to 6 for each element of Q until the STACK is empty.3.If an operand is encountered, add it to P4.If a left parenthesis is encountered, Push it onto STACK. Push it onto STACK.5.If an operator × is encountered, then: (a)Repeatedly pop from STACK and add to P each operator (on the top of STACK) which has the same precedence as or higher precedence than × (b)Add × to STACK. [End of If structure.]6.If a right parenthesis is encountered, then: (a) Repeatedly pop from STACK and add to P each operator (on the top of STACK) until a left parenthesis is encountered. (b)Remove the left parenthesis. [Do not add the left parenthesis to P.][End of If structure.][End of Step 2 loop]7.Exit

**PROGRAM: [Write program to convert infix expression to postfix expression]**

**OUTPUT:**

**CONCLUSION:**