Data Structures Odyssey: Exploring the Foundations of Computing

Ex. No.: 05	Infix to Postfix Conversion	Date:28/03/2024

Write a C program to perform infix to postfix conversion using stack.

Algorithm:

1) Start
2) Initialize an empty stack to hold operators.
3) Initialize an empty string to store the postfix expression.
4) Iterate through each character in the infix expression:
a. If the character is an operand, add it to the postfix string.
b. If the character is an operator:
i. While the stack is not empty and the precedence of the top operator in the stack is greater than or equal to the current operator, pop operators from the stack and add them to the postfix string. ii. Push the current operator onto the stack.
c. If the character is an opening parenthesis '(', push it onto the stack.
d. If the character is a closing parenthesis ')':
i. Pop operators from the stack and add them to the postfix string until an opening parenthesis is encountered. ii. Discard the opening parenthesis.
5) Pop any remaining operators from the stack and add them to the postfix string. 6) Return the postfix expression.
7) Stop

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>

int top = 0; int stack[20];
char infix[40], postfix[40];

void convertToPostfix();
void push(int); char
pop();

int main() {     printf("Enter the infix
expression: ");     scanf("%s", infix);
convertToPostfix();     return 0;
}

void convertToPostfix() {     int
i, j = 0;     for (i = 0; infix[i] !=
'\0'; i++) {         switch (infix[i])
{            case '+':
               while (stack[top] >= 1)
postfix[j++] = pop();
push(1);                break;
case '-':               while
(stack[top] >= 1)
postfix[j++] = pop();

               push(2);
break;            case
'*':
               while (stack[top] >= 3)
postfix[j++] = pop();
```

```
push(3);                break;
case '/':
            while (stack[top] >= 4)
postfix[j++] = pop();
push(4);                break;
case '^':               postfix[j++] =
pop();              push(5);
break;          case '(':
push(0);                break;
case ')':
            while (stack[top] != 0)
postfix[j++] = pop();
top--;              break;
default:
            postfix[j++] = infix[i];
      }
   }
   while (top > 0)


         postfix[j++] = pop();
   printf("\nPostfix expression is =>\n\t\t%s", postfix);
}

void push(int element) {
top++;    stack[top] =
element;
}

char pop() {
   int el;    char
e;    el =
stack[top];
```

```
top--;     switch
(el) {       case
1:           e = '+';
break;       case
2:           e = '-';
break;       case
3:           e = '*';
break;       case
4:           e = '/';
break;       case
5:           e = '^';
break;
   }

return e;
}
```

OUTPUT:

```
Enter the infix expression: ab+cd+*ef/*g^

Postfix expression is =>
                abcd+ef/*g*^+aiml231501179@cselab:~$
```

RESULT: Thus, the program was successfully executed.