EXERCISE 12

Write a C program to implement the application of Stack (Notations)

AIM:

To implement stack application for expression conversion: Infix to Postfix notation using C.

ALGORITHM:

1. Initialize an empty stack for operators.
2. Scan the infix expression from left to right.
3. If the scanned character is an operand, add it to the postfix expression.
4. If it is an operator:
    - Pop from the stack to postfix while the stack is not empty and precedence of top of stack is greater or equal.
    - Push the scanned operator to the stack.
5. If it is '(', push to stack.
6. If it is ')', pop and output from the stack until '(' is found.
7. After the entire expression is scanned, pop all operators from the stack to postfix.

PROGRAM:

```
#include <stdio.h>

#include <ctype.h>

char stack[100];

int top = -1;

void push(char c) {

   stack[++top] = c;

}

char pop() {
```

```
    return stack[top--];
}
int precedence(char c) {
    if (c == '+' || c == '-') return 1;
    if (c == '*' || c == '/') return 2;
    return 0;
}
int main() {
    char infix[100], ch;
    int i = 0;
    printf("Enter infix expression: ");
    scanf("%s", infix);
    printf("Postfix expression: ");
    while (infix[i] != '\0') {
        ch = infix[i];
        if (isalnum(ch)) {
            printf("%c", ch); // print operands
        } else if (ch == '(') {
            push(ch);
        } else if (ch == ')') {
            while (stack[top] != '('
                printf("%c", pop());
            pop(); // remove '('
        } else { // operator
            while (top != -1 && precedence(stack[top]) >= precedence(ch))
                printf("%c", pop());
```

```
            push(ch);
        }
        i++;
    }
    while (top != -1)
        printf("%c", pop());
    return 0;
}
```

Input&Output:

```
Enter infix expression: A+B*C
Postfix expression: ABC*+

=== Code Execution Successful ===
```

RESULT :

The program executed successfully by implement stack application for expression conversion: Infix to Postfix notation using C.