**C Program**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define MAX\_SIZE 100

typedef struct {

char items[MAX\_SIZE];

int top;

} Stack;

void initialize(Stack \*s);

void push(Stack \*s, char c);

char pop(Stack \*s);

int precedence(char op);

int isOperator(char c);

void infixToPostfix(char \*infix, char \*postfix);

int main() {

char infix[MAX\_SIZE];

char postfix[MAX\_SIZE];

printf("Enter infix expression: ");

fgets(infix, MAX\_SIZE, stdin);

if (infix[strlen(infix) - 1] == '\n')

infix[strlen(infix) - 1] = '\0';

infixToPostfix(infix, postfix);

printf("Postfix expression: %s\n", postfix);

return 0;

}

void initialize(Stack \*s) {

s->top = -1;

}

void push(Stack \*s, char c) {

if (s->top == MAX\_SIZE - 1) {

printf("Stack Overflow\n");

exit(EXIT\_FAILURE);

}

s->items[++(s->top)] = c;

}

char pop(Stack \*s) {

if (s->top == -1) {

printf("Stack Underflow\n");

exit(EXIT\_FAILURE);

}

return s->items[(s->top)--];

}

int precedence(char op) {

switch (op) {

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

default:

return 0;

}

}

int isOperator(char c) {

return (c == '+' || c == '-' || c == '\*' || c == '/');

}

void infixToPostfix(char \*infix, char \*postfix) {

Stack stack;

initialize(&stack);

int i, j;

char token, topToken;

j = 0;

for (i = 0; infix[i] != '\0'; i++) {

token = infix[i];

if (isalnum(token)) {

postfix[j++] = token;

} else if (isOperator(token)) {

while (stack.top != -1 && precedence(token) <= precedence(stack.items[stack.top])) {

postfix[j++] = pop(&stack);

}

push(&stack, token);

} else if (token == '(') {

push(&stack, token);

} else if (token == ')') {

topToken = pop(&stack);

while (topToken != '(') {

postfix[j++] = topToken;

topToken = pop(&stack);

}

}

}

while (stack.top != -1) {

postfix[j++] = pop(&stack);

}

postfix[j] = '\0';

}

**Input:**

Enter infix expression: 7%4\*3+9

**Output:**

Postfix expression: 743\*9+





