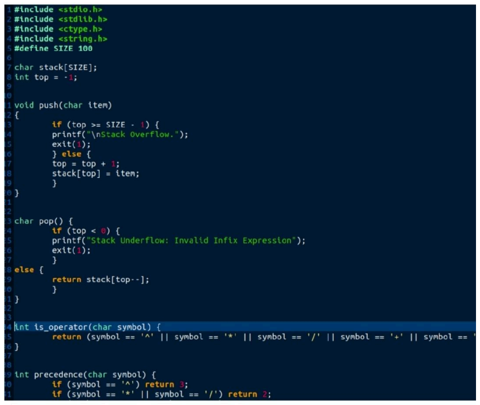
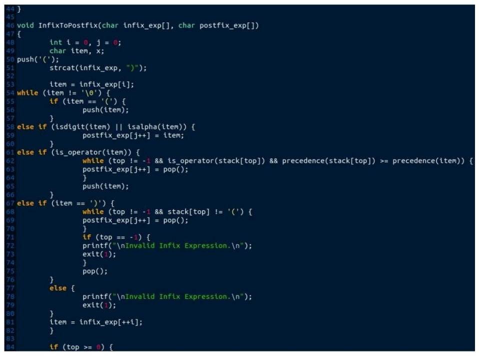
Submitted By:
Name of Student: Shamal Bhanudas Deore
Roll No.:18
Date of Performance:-16-08-24
Date of Submission:28-8-24

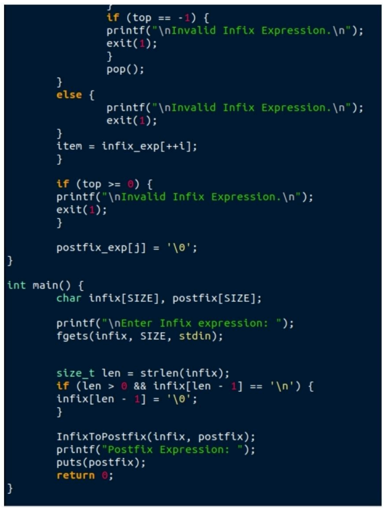
Code-



```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#define SIZE 100

char stack[SIZE];
int top = -1;


void push(char item)
{
        if (top >= SIZE - 1) {
        printf("\nStack Overflow.");
        exit(1);
        } else {
        top = top + 1;
        stack[top] = item;
        }
}


char pop() {
        if (top < 0) {
        printf("Stack Underflow: Invalid Infix Expression");
        exit(1);
        }
else {
        return stack[top--];
        }
}


int is_operator(char symbol) {
        return (symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+' || symbol == '
}


int precedence(char symbol) {
        if (symbol == '^') return 3;
        if (symbol == '*' || symbol == '/') return 2;
```

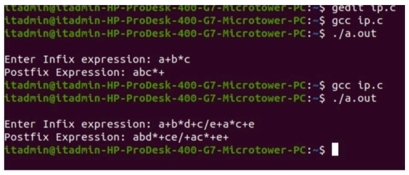


```c
}

void InfixToPostfix(char infix_exp[], char postfix_exp[])
{
        int i = 0, j = 0;
        char item, x;
push('(');
        strcat(infix_exp, ")");

        item = infix_exp[i];
while (item != '\0') {
        if (item == '(') {
                push(item);
        }
else if (isdigit(item) || isalpha(item)) {
                postfix_exp[j++] = item;
        }
else if (is_operator(item)) {
                while (top != -1 && is_operator(stack[top]) && precedence(stack[top]) >= precedence(item)) {
                postfix_exp[j++] = pop();
                }
                push(item);
        }
else if (item == ')') {
                while (top != -1 && stack[top] != '(') {
                postfix_exp[j++] = pop();
                }
                if (top == -1) {
                printf("\nInvalid Infix Expression.\n");
                exit(1);
                }
                pop();
        }
        else {
                printf("\nInvalid Infix Expression.\n");
                exit(1);
        }
        item = infix_exp[++i];
        }

        if (top >= 0) {
```



```c
                if (top == -1) {
                printf("\nInvalid Infix Expression.\n");
                exit(1);
                }
                pop();
        }
        else {
                printf("\nInvalid Infix Expression.\n");
                exit(1);
        }
        item = infix_exp[++i];
        }

        if (top >= 0) {
        printf("\nInvalid Infix Expression.\n");
        exit(1);
        }

        postfix_exp[j] = '\0';
}

int main() {
        char infix[SIZE], postfix[SIZE];

        printf("\nEnter Infix expression: ");
        fgets(infix, SIZE, stdin);


        size_t len = strlen(infix);
        if (len > 0 && infix[len - 1] == '\n') {
        infix[len - 1] = '\0';
        }

        InfixToPostfix(infix, postfix);
        printf("Postfix Expression: ");
        puts(postfix);
        return 0;
}
```

Output-



```
itadmin@itadmin-HP-ProDesk-400-G7-Microtower-PC:~$ gedit ip.c
itadmin@itadmin-HP-ProDesk-400-G7-Microtower-PC:~$ gcc ip.c
itadmin@itadmin-HP-ProDesk-400-G7-Microtower-PC:~$ ./a.out

Enter Infix expression: a+b*c
Postfix Expression: abc*+
itadmin@itadmin-HP-ProDesk-400-G7-Microtower-PC:~$ gcc ip.c
itadmin@itadmin-HP-ProDesk-400-G7-Microtower-PC:~$ ./a.out

Enter Infix expression: a+b*d+c/e+a*c+e
Postfix Expression: abd*+ce/+ac*+e+
itadmin@itadmin-HP-ProDesk-400-G7-Microtower-PC:~$ 
```