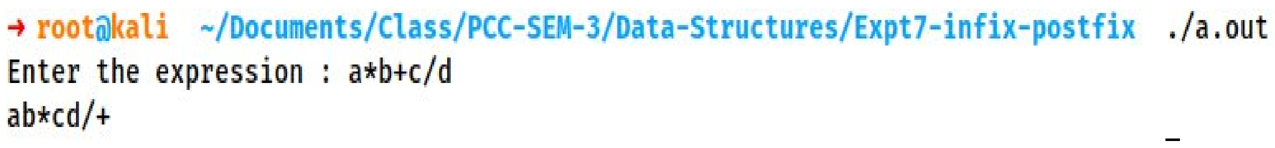
Program: Infix to Postfix

```
#include <stdio.h>
#define SIZE 20
int TOP = 0;
int STACK[SIZE];
int isSTACKFull(){
    if (TOP == SIZE)
        return 1;
    return 0;
}
void push(int val){
    if (isSTACKFull()){
        printf("Stack is Full. \n");
        return;
    }
    STACK[TOP++] = val;
}
int isSTACKEmpty(){
    if (TOP == 0)
        return 1;
    return 0;
}
int pop(){
    if (isSTACKEmpty()){
        printf("Stack is Empty. \n");
        return -1;
    }
    return STACK[--TOP];
}
int getTopStack(){
    if (isSTACKEmpty())
        return -1;
    return STACK[TOP - 1];
```

```
}
int getPrecedenceOfOperator(char c, int on_stack){
    switch (c){
    case '+':
    case '-':
        return 1;
    case '*':
    case '/':
        return 2;
    case '^':
        if (on_stack)
            return 9;
        else
            return 10;
    case '(':
        if (on_stack)
            return 0;
        else
            return 20;
    default:
        return -1;
    }
}
void inFixToPostFix(char s[]){
    int i = 0;
    while (s[i] != '\0'){
        char x = s[i], tmp;
        if ((x >= 65 && x <= 90) || (x >= 97 && x <= 122))
            printf("%c", x);
        else{
            if (x == ')'){
                while ((tmp = pop()) != '(')
                    printf("%c", tmp);
                i++;
                continue;
            }
            while (getPrecedenceOfOperator(x, 0) <=
                   getPrecedenceOfOperator(getTopStack(), 1))
                printf("%c", pop());
            push(x);
        }
        i++;
    }
    while (!isSTACKEmpty())
        printf("%c", pop());
    printf("\n");
}
int main(){
    char exp[100];
    printf("Enter the expression : ");
    scanf("%s", exp);
    inFixToPostFix(exp);
    return 0;
}
```

OUTPUT:



```
➜ root@kali  ~/Documents/Class/PCC-SEM-3/Data-Structures/Expt7-infix-postfix  ./a.out
Enter the expression : a*b+c/d
ab*cd/+
```