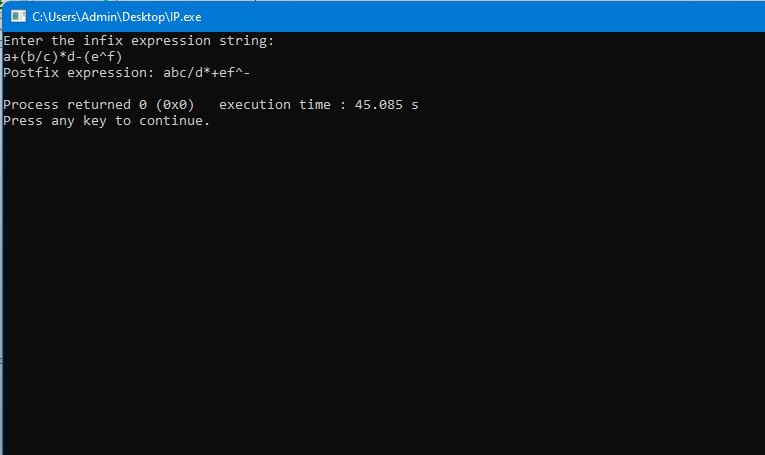
**WEEK 2 DS LAB:**

**Conversion of infix to postfix:**

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
  
#define MAX 20  
  
char s[MAX];  
int top = -1;  
  
int prec(char c) {  
    if (c == '^')  
        return 5;  
    else if (c == '/')  
        return 4;  
    else if (c == '\*')  
        return 3;  
    else if (c == '+')  
        return 2;  
    else if (c == '-')  
        return 1;  
    else  
        return -1;  
}  
  
int isEmpty() {  
    return top == -1;  
}  
  
int isFull() {  
    return top == MAX - 1;  
}  
  
char peek() {  
    return s[top];  
}  
  
char pop() {  
    if (isEmpty()) {  
        printf("Stack is empty\n");  
        return -1;  
    }  
    else  
    {  
        char ch = s[top];  
        top--;  
        return ch;  
    }  
}  
  
void push(char opr) {  
    if (isFull()) {  
        printf("Stack is full\n");  
    } else {  
        top = top + 1;  
        s[top] = opr;  
    }  
}  
  
void infixToPostfix(char infix[], char postfix[]) {  
    int i, j;  
    char ch;  
    for (i = 0, j = 0; i < strlen(infix); i++) {  
        ch = infix[i];  
        if ((ch >= '0' && ch <= '9') || (ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z')) {  
            postfix[j++] = ch;  
        } else if (ch == '(') {  
            push(ch);  
        } else if (ch == ')') {  
            while (!isEmpty() && peek() != '(') {  
                postfix[j++] = pop();  
            }  
            if (!isEmpty() && peek() != '(') {  
                printf("Invalid expression\n");  
                return;  
            } else {  
                pop();  
            }  
        } else {  
            while (!isEmpty() && prec(ch) <= prec(peek())) {  
                postfix[j++] = pop();  
            }  
            push(ch);  
        }  
    }  
  
    while (!isEmpty()) {  
        postfix[j++] = pop();  
    }  
  
    postfix[j] = '\0'; // Null-terminate the postfix expression  
}  
  
int main() {  
    char infix[20], postfix[20];  
    printf("Enter the infix expression string: \n");  
    scanf("%s", infix);  
    infixToPostfix(infix, postfix);  
    printf("Postfix expression: %s\n", postfix);  
    return 0;  
}

Output:



**Evaluation of Postfix Expression:**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

int n,top=-1,k;

int val,res,ans;

int stack[100];

char expn[100];

void push(int [],int);

int pop(int []);

int evaluate(char expn[]);

void main()

{

printf("Enter a postfix expression: \n");

scanf("%s",expn);

ans=evaluate(expn);

printf("Answer of above expression: \n");

printf("%d",ans);

}

int evaluate(char expn[])

{

int op1,op2,i=0;

n=strlen(expn);

while(expn[i]!='\0')

{

if(isdigit(expn[i]))

push(stack,(int)(expn[i]-'0'));

else

{

op1=pop(stack);

op2=pop(stack);

switch(expn[i])

{

case '+':

res=op1+op2;

break;

case '-':

res=op1-op2;

break;

case '\*':

res=op1\*op2;

break;

case '%':

res=op2%op1;

break;

case '/':

res=op2/op1;

break;

default:

continue;

}

if(res<0)

res=res\*(-1);

push(stack,res);

}

i++;

}

return stack[top];

}

void push(int stack[],int val)

{

if(top==n-1)

printf("Overflow");

else

{

top+=1;

stack[top]=val;

}

}

int pop(int stack[])

{

if(top==-1)

printf("Underflow");

else

{

val=stack[top];

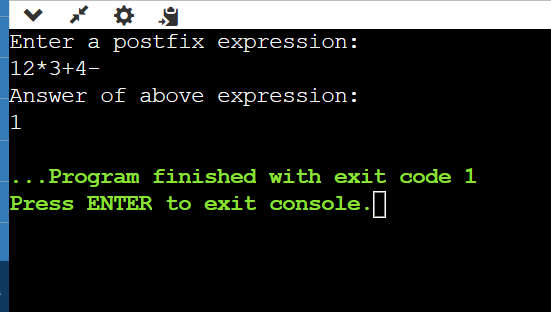
top-=1;

return val;

}

}

Output:

****