LAB 2 PROGRAM

1. Write a program to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply), / (divide) and ^ (power).

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX 100

void push(char st[],char ch);

char pop(char st[]);

void infix_to_postfix(char src[],char ans[]);

int isalpha_numeric(char ch);

int isOperator(char ch);

int isPrior(char ch);

int top = -1;

char st[MAX];

int main(){

char postfix[100],infix[100];

printf("Enter the infix expression\n");

scanf("%s",infix);

strcpy(postfix,"");

infix_to_postfix(infix,postfix);

printf("The postfix expression is\n");

printf("%s\n",postfix);
```

```
}

int isalpha_numeric(char ch){

if((ch>= 'a' && ch<='z')||(ch >='A' && ch <= 'Z')||(ch >= '0' &&

ch <= '9')){

return 1;

}else{

return 0;

}

}

int isOperator(char ch){

if(ch == '+' || ch == '-' || ch == '*' || ch == '/' ||ch == '%' ){

return 1;

}else{

return 0;

}

}

int isPrior(char ch){

if( ch == '*' || ch == '/' ||ch == '%'){

return 1;

}else{

return 0;

}

}

void infix_to_postfix(char src[],char ans[]){

int i=0;
int j =0;

while(src[i]!='\0') {

if(src[i] == '('){

push(st,src[i]);

}

else if(isalpha_numeric(src[i])){

ans[j]= src[i];

++j;

}

else if(isOperator(src[i])){

while(top != -1 && st[top] != '(' && (isPrior(st[top]) >=

isPrior(src[i]))){

ans[j] = pop(st);

++j;

}

push(st,src[i]);

}else if(src[i] == ')'){

while(top != -1 && st[top] != '('){

ans[j]= pop(st);

++j;

}

pop(st);

}

else{

printf("invalid expression");
```

```
exit(0);

}

++i;

}

while(top != -1 && st[top] != '('){

ans[j] = pop(st);

++j;

}

ans[j]='\0';

}

void push(char st[],char ch){

if(top == MAX-1){

printf("Stack overflow\n");

}

else{

++top;

st[top] = ch;

}

}

char pop(char st[]){

char ch = '\0';

if(top ==-1){

printf("Stack underflow\n");

}

else{
```

```
ch = st[top];

--top;

}

return ch;

}
```

OUTPUT:

```
Enter the infix expression

(a+b/c*(d+e)-f)

The postfix expression is

abc/de+*+f-
```