```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

struct node
{
    char info;
    struct node *left;
    struct node *right;
};

typedef struct node *NODE;

struct stack
{
    int top;
    NODE data[10];
};

typedef struct stack STACK;

int preced(char item){
    switch(item){
        case '^': return 5;
        case '*':
        case '/': return 3;
        case '+':
        case '-': return 1;
    }
}

void preorder(NODE root){
    if(root != NULL){
        printf("%c\t", root->info);
        preorder(root->left);
        preorder(root->right);
    }
}

void inorder(NODE root){
    if(root != NULL){
        inorder(root->left);
        printf("%c\t", root->info);
        inorder(root->right);
    }
}

void postorder(NODE root){
    if(root != NULL){
        postorder(root->left);
        postorder(root->right);
        printf("%c\t", root->info);
    }
}


void push(STACK *s, NODE temp){
    s->data[++(s->top)] = temp;
}

NODE pop(STACK *s){
    return (s->data[(s->top)--]);
}

NODE createnode(char item)
{
    NODE temp;
    temp = (NODE)malloc(sizeof(struct node));
    temp->info = item;
    temp->left = NULL;
    temp->right = NULL;
    return temp;
}

NODE createExpTree(char expr[20])
{
    STACK tree, operator;
    tree.top = -1;
    operator.top = -1;
    char symbol;
    int i;
    NODE temp, t, l, r;

    for (i=0; expr[i] != '\0'; i++)
    {
        symbol = expr[i];
        temp = createnode(symbol);
        if(isalnum(symbol))
```

```
            push(&tree, temp);
        else{
            if(operator.top == -1)
                push(&operator, temp);
            else{
                while(operator.top != -1 && preced((operator.data[operator.top])->info) >=
preced(symbol))
                {
                    t = pop(&operator);
                    r = pop(&tree);
                    l = pop(&tree);
                    t->right = r;
                    t->left = l;
                    push(&tree, t);
                }
                push(&operator, temp);
            }
        }
    }

    while(operator.top != -1){
      t = pop(&operator);
      r = pop(&tree);
      l = pop(&tree);
      t->right = r;
      t->left = l;
      push(&tree, t);
    }

  return pop(&tree);
}




int main()
{
    NODE root = NULL;
    char expr[20];
    printf("Read expression\n");
    scanf("%s", expr);
    root = createExpTree(expr);
    printf("\nInorder::");
    inorder(root);
    printf("\nPreorder:");
    preorder(root);
    printf("\nPostorder:");
    postorder(root);
    return 0;
}
```