```c
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
typedef char element;
struct arrstack
{
    element *a;
    int m;
    int top;
};
typedef struct arrstack * stack;
stack creatememory(int);
void push(stack,element);
element pop(stack);
element topelement(stack);
int isempty(stack);
int isfull(stack);

void infixtopostfix(char *,char *);
int precedence(char ch);


int main()
{
    char *infix,*postfix;
    infix=(char *)malloc(sizeof(char)*20);
    postfix=(char *)malloc(sizeof(char)*20);
    printf("\nEnter an infix expression:");
    gets(infix);
    infixtopostfix(infix,postfix);
    printf("\nThe postfix expression is:");
    puts(postfix);
    return 0;
}
stack creatememory(int sz)
{
    stack s;
    s=(stack)malloc(sizeof(struct arrstack));
    s->a=(element *)malloc(sizeof(element)*sz);
    s->m=sz;
    s->top=0;
    return s;
}
int isempty(stack s)
{
    if(s->top==0)
        return 1;
    return 0;
}
int isfull(stack s)
{
    if(s->top==s->m)
        return 1;
    return 0;
}
void push(stack s,element e)
{
    if(!isfull(s))
        s->a[s->top++]=e;
    else
        printf("\nStack Overflow");
}
element pop(stack s)
{
    if(!isempty(s))
        return(s->a[--s->top]);
    else
        printf("\nStack Underflow");

}
element topelement(stack s)
{
    return(s->a[s->top-1]);
}
void infixtopostfix(char *infix,char *postfix)
{
    int i=0,j=0;
    char ch;
```

```c
    stack s;
    s=creatememory(20);
    push(s,'$');
    puts(infix);
    while(infix[i]!='\0')
    {
        ch=infix[i++];
        if(isalpha(ch))
            postfix[j++]=ch;
        else
        {

while(precedence(topelement(s))>=precedence(ch))
                postfix[j++]=pop(s);
            push(s,ch);
        }
    }
    while(topelement(s)!='$')
        postfix[j++]=pop(s);
    postfix[j]='\0';
}
int precedence(char ch)
{
    if(ch=='+'||ch=='-')
        return 1;
    else if(ch=='*'||ch=='/')
        return 2;
    else if(ch=='^')
        return 3;
    else if(ch=='$')
        return 0;
}
```