```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);

char * infixtopostfix(char *);
int precedence(char ch);
char typeofch(char);


int main()
{
    //char infix[100],postfix[100]; //static
    char *infix,*postfix;
    infix=(char *)malloc(sizeof(char)*50);
    printf("\nEnter an infix expression:");
    gets(infix);
    postfix=infixtopostfix(infix);
    printf("\nThe postfix expression is:");
    puts(postfix);
    return 0;
}
stack creatememory(int sz)
{
    stack s;
    s=(stack)malloc(sizeof(struct arrstack));
```

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

```c
    char ch,t,*postfix;
    stack s;
    s=creatememory(50);
    postfix=(char *)malloc(sizeof(char)*20);
    push(s,'(');
    while((ch=infix[i++])!='\0')
    {
        t=typeofch(ch);
        switch(t)
        {
            case '(':
                push(s,ch);
                break;
            case ')':
                while(topelement(s)!='(')
                    postfix[j++]=pop(s);
                pop(s);
                break;
            case '1':
                postfix[j++]=ch;
                break;
            case '2':

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