```
#include<stdio.h>
#include<stdlib.h>
typedef int element;
struct node
{
    element data;
    struct node *next;
};
typedef struct node *stack;
typedef struct node *position;


stack makenullstack();
void push(stack,element);
element pop(stack);
int isempty(stack);
position firstpos(stack);
position toppos(stack);
position nextpos(stack,position);
position prevpos(stack,position);
void printstack(stack);

int main()
{
    int i=0,j=0,c=0,n1,n2;
    char *postfixwithop,*postfixwithlit,ch;
    int opvalue;
    stack s;
    s=makenullstack();
    postfixwithop=(char
*)malloc(sizeof(char)*50);
    postfixwithlit=(char
*)malloc(sizeof(char)*50);
    printf("\nEnter postfix expression with
operands:\n");
    scanf("%s",postfixwithop);
    while((ch=postfixwithop[i++])!='\0')
    {
        if(isalpha(ch))
        {
            printf("Enter the value of %d 
operand:",++c);
            scanf("%d",&opvalue);
            postfixwithlit[j++]=opvalue+'0';
        }
        else
            postfixwithlit[j++]=ch;
    }
    postfixwithlit[j]='\0';
    printf("%d operands are present:\n",c);
    printf("Postfix expression with literals 
is:%s",postfixwithlit);
    j=0;
    while((ch=postfixwithlit[j++])!='\0')
    {
        if(ch>='0'&&ch<='9')
        {
            push(s,ch-'0');
        }
        else if(ch=='+')
        {
            n2=pop(s);
            n1=pop(s);
            push(s,n1+n2);
        }
        else if(ch=='-')
        {
            n2=pop(s);
            n1=pop(s);
            push(s,n1-n2);
        }
        else if(ch=='*')
        {
            n2=pop(s);
            n1=pop(s);
            push(s,n1*n2);
        }
        else if(ch=='/')
```

```c
            {
                n2=pop(s);
                n1=pop(s);
                push(s,n1/n2);
            }

        }
    if(!isempty(s))
        printf("\nResult is:%d\n",pop(s));
}
stack makenullstack()
{
    stack s;
    s=(stack)malloc(sizeof(struct node));
    s->next=NULL;
    return s;
}
void push(stack s,element e)
{
    position p=toppos(s);
    stack t;
    t=makenullstack();
    t->data=e;
    p->next=t;
}
element pop(stack s)
{
    element e;
    position p=toppos(s);
    position q=prevpos(s,p);
    e=p->data;
    q->next=NULL;
    free(p);
    return e;
}
int isempty(stack s)
{
    if(s->next==NULL)
        return 1;
    return 0;
}
position firstpos(stack s)
{
    return s;
}
position toppos(stack s)
{
    position p=s;
    while(p->next!=NULL)
        p=p->next;
    return p;
}
position nextpos(stack s,position p)
{
    return p->next;
}
position prevpos(stack s,position p)
{
    position q=firstpos(s);
    while(q->next!=p)
        q=q->next;
    return q;
}
void printstack(stack s)
{
    position i;

for(i=firstpos(s);i!=toppos(s);i=nextpos(s,i))
        printf("%d ",i->next->data);
}
```