```
//M Gayathri-185001050
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<ctype.h>
#include"stack.h"

int main()
{
 int flag=1,l;
 char in[20],post[20];
 printf("enter an expression");
 scanf("%s",in);
 postfix(in);

return 0;
}

/*sample input/output

enter an expression7-(((3+2)*(6+1))/(5+6)

unbalanced
enter an expression(((3+2)*(2+5)

unbalanced
enter an expression((3+2)*6/7)

32+67/*
4.285714

*/

/******************CONTENTS OF FILE STACK.H*****************/
/*

      void evaluate(char post[])
{

      struct node
      {
      float data;
      struct node *next;
      }*topf=NULL;

      void display()
      {
      printf("\nCONTENTS ARE:-");
      for( struct node *temp=topf; temp!=NULL ; temp=temp->next)
    printf("%f\t", temp->data  );
      }

      void push(float x)
      {
```

```
  struct node *new;
  new=(struct node*)malloc(sizeof(struct node));
  new->data=x;
  if(topf==NULL)
        new->next=NULL;
  else
        new->next=topf;
  topf=new;
  }

  void pop()
  {
  if(topf==NULL)
  printf("\nstack is empty");
  else
  {
struct node *temp;
  temp=(struct node*)malloc(sizeof(struct node));
temp=topf;
topf=topf->next;
free(temp);
  }

  }
  float peek()
  {
  if(topf==NULL)
  {
  printf("\nStack is empty");
  return 0;
  }
  else
  {
  return topf->data;
  }
  }

  int l1=strlen(post);
  float t1,t2,ans,f;
  char c;
  for(int k=0;k<l1;k++)
  {
  if(post[k]>='0' && post[k]<='9')
  {
        c=post[k];
        f=(float)c-48.0;
        push(f);
  }
  else
        {
        if(post[k]=='+')
        {
              t1=peek();
              pop();
```

```
                t2=peek();
                pop();
                ans=t2+t1;
                push(ans);
          }
          else if(post[k]=='-')
          {
                t1=peek();
                pop();
                t2=peek();
                pop();
                ans=t2-t1;
                push(ans);
          }
          else if(post[k]=='*')
          {
                t1=peek();
                pop();
                t2=peek();
                pop();
                ans=t2*t1;
                push(ans);
          }
          else
          {
                t1=peek();
                pop();
                t2=peek();
                pop();
                ans=t2/t1;
                push(ans);
          }
      }
      }
      printf("\n%f",ans);
}

struct node
{
 char data;
 struct node *next;
}*top=NULL;

void display()
{
  printf("CONTENTS ARE:-");
  for( struct node *temp=top; temp!=NULL ; temp=temp->next)
    printf("%c\t", temp->data  );
  printf("\n");

}

void push(char x)
{
```

```
      struct node *new;
      new=(struct node*)malloc(sizeof(struct node));
      new->data=x;
      if(top==NULL)
            new->next=NULL;
      else
            new->next=top;
      top=new;
}


void pop()
{
 if(top==NULL)
  printf("\nstack is empty");
 else
  {
    struct node *temp;
      temp=(struct node*)malloc(sizeof(struct node));
    temp=top;
    top=temp->next;
   free(temp);
  }

}
char peek()
{
 if(top==NULL){
  //printf("\nStack is empty");
  return 0;}
 else
  return top->data;
}



void postfix(char in[])
{
 int l=strlen(in);
 int j=0;
 char post[20];



 int flag=0;
 for(int k=0;k<l;k++)
 {
      if(in[k]=='(')
            push(in[k]);
      else
      {
            if(peek()=='(' && in[k]==')')
                  {
                        if(peek())
```

```
                                pop();
                        }
            }
}
if(top==NULL)
{
flag=1;
 }
 else
 printf("\nunbalanced");
 if(flag==1)
 {
 for(int i=0;i<l;i++)
 {
    if(in[i]=='(')
      push('(');
    else if(in[i]==')')
    {
      while(peek()!='(')
        {
          post[j]=peek();
          j++;
          pop();
        }
      pop();
    }
    else if(in[i]=='+' || in[i]=='-')
    {
      if((peek()=='+')||(peek()=='-')||(peek()=='*')||(peek()=='/' ))
      {
        while(peek()!='(')
            {
                  post[j]=peek();
            j++;
            pop();
            }
            push(in[i]);
      }
      else
        push(in[i]);
    }
  else if((in[i]=='*')||(in[i]=='/'))
  {
   if((peek()=='*')||(peek()=='/'))
   {
    while((peek()=='+')||(peek()=='-')||(peek()=='('))
    {
      post[j]=peek();
      j++;
      pop();
      }
      push(in[i]);
   }
   else
```

```
    push(in[i]);
  }
  else
   {
    post[j]=in[i];
    j++;
   }

  }

 printf("\n%s",post);
 evaluate(post);}
}
*/
```