```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int top=1,i=0,l=0;
int flag=1;
char stack[100]="$E",inp[100],t[100];
char table[11][6][10]={{"TU","e1","e1","TU","e1","e1"},
                {"e1","+TU","e1","e1","#","#"},
                {"FV","e1","e1","FV","e1","e1"},
                {"e1","#","*FV","e1","#","#"},
                {"i","e1","e1","(E)","e1","e1"},
                {"pop"," "," "," "," "," "},
                {" ","pop"," "," "," "," "},
                {" "," ","pop"," "," "," "},
                {" "," "," ","pop"," "," "},
                {"e2","e2","e2","e2","pop","e2"},
                {"e3","e3","e3","e3","e3","accept"}};

void pop()
{
   stack[top--]='\0';
}
int terminal(char a)
{
   if(a=='i'||a=='+'||a=='*'||a=='('||a==')'||a=='$')
      return 1;
   else
      return 0;
}
void rev(char *s)
{
   int j;
   char b;
   for(j=0;j<strlen(s)/2;j++){
      b=s[j];
      s[j]=s[strlen(s)-j-1];
      s[strlen(s)-j-1]=b;
   }
}
int get_stack(char a)
{
   if(a=='E')
      return 0;
   else if(a=='U')
      return 1;
   else if(a=='T')
      return 2;
   else if(a=='V')
      return 3;
   else if(a=='F')
      return 4;
   else if(a=='i')
      return 5;
   else if(a=='+')
      return 6;
   else if(a=='*')
      return 7;
   else if(a=='(')
      return 8;
   else if(a==')')
      return 9;
   else if(a=='$')
      return 10;
}
int get_inp(char a)
{
   if(a=='i')
      return 0;
   else if(a=='+')
      return 1;
   else if(a=='*')
      return 2;
   else if(a=='(')
      return 3;
   else if(a==')')
      return 4;
```

```
    else if(a=='$')
        return 5;
}
void outputS()
{
    int k;
    printf("\n");
    for(k=0;k<=top;k++)
        printf("%c",stack[k]);
}
void outputI()
{
    int k;
    printf("\t\t");
    for(k=i;k<l;k++)
        printf("%c",inp[k]);
}
void parse()
{
    int k,f=0;
    printf("\nSTACK\t\tINPUT\t\tMESSAAGE");
    char X,a;
    outputS();
    outputI();
    while(1)
    {
        X=stack[top];
        a=inp[i];
        if(strcmp(table[get_stack(X)][get_inp(a)],"e1")==0)
        {
            f=1;
            printf("\t\tMISSING OPERAND : add 'i' onto input");
            flag=0;
            for(k=l;k>=i;k--)
                inp[k+1]=inp[k];
            inp[k+1]='i';
            l++;
            outputS();
            outputI();
        }
        else if(X=='$' && a=='$')
        {
          if(flag) printf("\nACCEPT");
          else printf("\nPARSED STRING NOT ACCEPTED");
            if(f==1)
                printf("\nRECOVERED STRING : %s",inp);
            return;
        }
        else if(X=='$')
        {
            f=1;
            printf("\t\tUNEXPECTED %c ",a);
            inp[i]='$';
            inp[i+1]='\0';
            l=i+1;
        }
        else if(X==')'&& a!=')')
        {
            f=1;
            printf("\t\tMISSING RIGHT PARENTHESIS");
            for(k=l;k>=i;k--)
                inp[k+1]=inp[k];
            inp[k+1]=')';
            l++;
            outputS();
            outputI();
        }
        else if(X==a)
        {
            pop();
            i++;
            outputS();
            outputI();
        }
        else
        {
            pop();
            strcpy(t,table[get_stack(X)][get_inp(a)]);
            rev(t);
            if(strcmp(t,"#")!=0)
            {
               strcat(stack,t);
               top=top+strlen(t);
            }
            outputS();
            outputI();
            printf("\t\t%c->%s",X,table[get_stack(X)][get_inp(a)]);
        }
    }
}

int main()
{
    printf("GRAMMER :");
    printf("\nE->TU");
    printf("\nU->+TU|#");
    printf("\nT->FV");
    printf("\nV->*FV|#");
    printf("\nF->(E)|i");
    printf("\n\nwhere U stands for E', V stands for T', i stands for Id and # stands for NULL\n");
    printf("\nEnter the string to be parsed\n\n");
    gets(inp);
    printf("\n\n");
    l=strlen(inp);
    inp[l]='$';
    inp[l+1]='\0';
    l++;
    parse();
    return 0;
}


/*
GRAMMER :
E->TU
U->+TU|#
T->FV
V->*FV|#
F->(E)|i

where U stands for E', V stands for T', i stands for Id and # stands for NULL

Enter the string to be parsed
i+i*i+i+i

STACK      INPUT      MESSAAGE
$E      i+i*i+i+i$
$UT        i+i*i+i+i$      E->TU
$UVF       i+i*i+i+i$      T->FV
$UVi       i+i*i+i+i$      F->i
$UV        +i*i+i+i$
$U      +i*i+i+i$      V->#
$UT+       +i*i+i+i$      U->+TU
$UT        i*i+i+i$
$UVF       i*i+i+i$      T->FV
$UVi       i*i+i+i$      F->i
$UV        *i+i+i$
$UVF*        *i+i+i$      V->*FV
$UVF       i+i+i$
$UVi       i+i+i$      F->i
$UV        +i+i$
$U      +i+i$      V->#
$UT+       +i+i$      U->+TU
$UT        i+i$
$UVF       i+i$        T->FV
$UVi       i+i$        F->i
$UV        +i$
$U      +i$      V->#
$UT+       +i$      U->+TU
$UT        i$
$UVF       i$        T->FV
$UVi       i$        F->i
$UV        $
$U      $      V->#
```

```
$    $    U->#
ACCEPT
*/

```