```
/* ///*
 * ASSIGNMENT: LL(1) PARSER
 * AUTHORS: BIJAY KHATRI & DEEPAK VERMA
 * ROLL: 12/CS/45 AND 12/CS/46
 *//// */
#include <stdio.h>
#include <string.h>
#include <unistd.h>
char input[10];
int precedence[256][256];
int i=0;
int lim;
int top=-1;
char stack[60];
char grammar[10][10];
void push(char symbol)
{  stack[++top]=symbol;     }
char pop()
{    char val=stack[top];       top--;    return val;    }
void display()
{    int jm;
   char stk_str[20];
   for(jm=0;jm<=top;jm++)   printf("%c",stack[jm]);
}
int is_operator(char op)
{    return op=='+' || op=='-' || op=='*' || op=='/' || op=='^';      }
void getLine(FILE *fp)
{
   int j=0;
   char ch;
   scanf("%d",&lim);
   while(i<lim)
   {
      ch=getchar();
      if(ch==EOF)
         break;

      if(ch=='\n')
      {    i++,j=0;       }
      else
      {         grammar[i][j]=ch;        j++;           }
   }
   scanf("%s",input);
}

int main(void)
{
   precedence['+']['+']=1;
   precedence['+']['-']=1;
   precedence['+']['*']=-1;
   precedence['+']['/']=-1;
   precedence['+']['$']=1;

   precedence['-']['+']=1;
   precedence['-']['-']=1;
   precedence['-']['*']=-1;
   precedence['-']['/']=-1;
   precedence['-']['$']=1;

   precedence['*']['+']=1;
   precedence['*']['-']=1;
   precedence['*']['*']=1;
   precedence['*']['/']=1;
   precedence['*']['$']=1;

   precedence['/']['+']=1;
   precedence['/']['-']=1;
   precedence['/']['*']=1;
   precedence['/']['/']=1;
   precedence['/']['$']=1;

   getLine(stdin);
    int m,n;
    strcat(input,"$");
    int index=0;
    int len= strlen(input);
    push('$');
    int flag=1;
    printf("STACK\t  INPUT BUFFER  \tACTION\n");
    while(index<len && flag)
   {        char current= input[index];
       if(stack[top]=='$' && current!='$')
      {
         printf("\n");
         display();
         printf("\t\t%s\t\tSHIFT %c\n",input+index,current);
         push(current);
         index++;
      }

      else if(top>0)
      {        if(top==1 && stack[top]==grammar[1][0] && current=='$')
         {
            display();
            printf("\t\t%s\t\tACCEPT\n",input+index);
            break;
         }
       else if(stack[top]==current)
         {
            display();
            char str[20];
            if(is_operator(current))
               strcpy(str,"operand");
            else
               strcpy(str,"operator");

            printf("\t\t%s\t\tError.....%s missing.....recovering..\n",input+index,str);
            index++;
         }

         else if(top%2==1 && stack[top]>='a' && stack[top]<='z')
            {
               display();
               char val_on_top = pop();
               int temp;
               int f=0;
               for(temp=0; temp<lim;temp++)
               {
                  if(strlen(grammar[temp])==3 && grammar[temp][2]==val_on_top)
                     {
                        push(grammar[temp][0]);
                        printf("\t\t%s\t\tREDUCE BY %s\n",input+index, grammar[temp]);
                        f=1;
                        break;
                     }
               }
               if(f==0)
               {
                  flag=0;
                  printf("\t\t%s\t\t",input+index);
                  printf("REJECTED\n");
                  break;
               }
            }

         else if(top%2==0 && current!='$')
         {
            display();
            printf("\t\t%s\t\tSHIFT %c\n",input+index, current);
            push(current);
            index++;
         }

         else if((top>=3&& top%2==1) && stack[top]>='A' && stack[top]<='Z' && precedence[stack[top-1]][current]==1)
         {
            display();
            char val1= pop();
            char val2= pop();
            char val3= pop();
               int temp;
               int f=0;
               for(temp=0; temp<lim;temp++)
               {
                  if(strlen(grammar[temp])==5 && grammar[temp][2]==val1 && grammar[temp][3]==val2 && grammar[temp][4]==
val3)
                    {
                        push(grammar[temp][0]);
                        printf("\t\t%s\t\tREDUCE BY %s\n",input+index,grammar[temp]);
                        f=1;
                        break;
                    }
                }
                if(f==0)
                {
                    flag=0;
                    printf("\t\t%s\t\t",input+index);
                    printf("REJECTED\n");
                    break;
                }

            }
            else if(top==1 && stack[top]>='A' && stack[top]<='Z' || (top==3 && precedence[stack[top-1]][current]==-1))
            {
                display();
                printf("\t\t%s\t\tSHIFT %c\n",input+index,current);
                push(current);
                index++;
            }

            else if(current=='$' && stack[top]!=grammar[1][0])
            {
                if(is_operator(stack[top]))
                {

                    display();
                    printf("\t\t%s\t\tError occurred.... recovering... \n",input+index);
                    pop();
                }
                else
                {
                    display();
                    printf("\t\t%s\t\tREJECT\n",input+index);
                    break;
                }
            }
        }
    }
}



/***************INPUT*************************

   5
   E=(E)
   E=E+E
   E=E*E
   E=a
   a++aaa+

**************************************************/


/*****************OUTPUT**********************

   STACK      INPUT BUFFER      ACTION

   $        a++aaa+$        SHIFT a
   $a          ++aaa+$         REDUCE BY E=a
   $E          ++aaa+$         SHIFT +
   $E+           +aaa+$        Error.....operand missing.....recovering..
   $E+           aaa+$          SHIFT a
   $E+a      aa+$         Error.....operator missing.....recovering..
   $E+a      a+$              Error.....operator missing.....recovering..
   $E+a      +$            REDUCE BY E=a
   $E+E      +$            REDUCE BY E=E+E
   $E        +$            SHIFT +
   $E+            $              Error occurred.... recovering...
   $E          $              ACCEPT

*************************************************/
```