```
/* ///*
 * 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,"$");
    int index=0;
    int len= strlen(input);
    push('$');
```

```
78    int flag=1;
79    printf("STACK\t  INPUT BUFFER  \tACTION\n");
80    while(index<len && flag)
81    {       char current= input[index];
82       if(stack[top]=='$' && current!='$')
83       {
84          printf("\n");
85          display();
86          printf("\t\t%s\t\tSHIFT %c\n",input+index,current);
87          push(current);
88          index++;
89       }
90
91       else if(top>0)
92       {       if(top==1 && stack[top]==grammar[1][0] && current=='$')
93          {
94             display();
95             printf("\t\t%s\t\tACCEPT\n",input+index);
96             break;
97          }
98        else if(stack[top]==current)
99          {
100            display();
101            char str[20];
102            if(is_operator(current))
103               strcpy(str,"operand");
104            else
105               strcpy(str,"operator");
106
107            printf("\t\t%s\t\tError.....%s missing.....recovering..\n",input+index,str);
108            index++;
109         }
110
111         else if(top%2==1 && stack[top]>='a' && stack[top]<='z')
112            {
113               display();
114               char val_on_top = pop();
115               int temp;
116               int f=0;
117               for(temp=0; temp<lim;temp++)
118               {
119                  if(strlen(grammar[temp])==3 && grammar[temp][2]==val_on_top)
120                     {
121                        push(grammar[temp][0]);
122                        printf("\t\t%s\t\tREDUCE BY %s\n",input+index, grammar[temp]);
123                        f=1;
124                        break;
125                     }
126               }
127               if(f==0)
128               {
129                  flag=0;
130                  printf("\t\t%s\t\t",input+index);
131                  printf("REJECTED\n");
132                  break;
133               }
134            }
135
136         else if(top%2==0 && current!='$')
137         {
138            display();
139            printf("\t\t%s\t\tSHIFT %c\n",input+index, current);
140            push(current);
141            index++;
142         }
143
144         else if((top>=3&& top%2==1) && stack[top]>='A' && stack[top]<='Z' && precedence[stack[top-1]][current]==1)
145         {
146            display();
147            char val1= pop();
148            char val2= pop();
149            char val3= pop();
150               int temp;
151               int f=0;
152               for(temp=0; temp<lim;temp++)
153               {
154                  if(strlen(grammar[temp])==5 && grammar[temp][2]==val1 && grammar[temp][3]==val2 && grammar[temp][4]==
```

```
    val3)
155                     {
156                        push(grammar[temp][0]);
157                        printf("\t\t%s\t\tREDUCE BY %s\n",input+index,grammar[temp]);
158                        f=1;
159                        break;
160                     }
161               }
162               if(f==0)
163               {
164                  flag=0;
165                  printf("\t\t%s\t\t",input+index);
166                  printf("REJECTED\n");
167                  break;
168               }
169
170         }
171         else if(top==1 && stack[top]>='A' && stack[top]<='Z' || (top==3 && precedence[stack[top-1]][current]==-1))
172         {
173            display();
174            printf("\t\t%s\t\tSHIFT %c\n",input+index,current);
175            push(current);
176            index++;
177         }
178
179         else if(current=='$' && stack[top]!=grammar[1][0])
180         {
181            if(is_operator(stack[top]))
182            {
183
184               display();
185              printf("\t\t%s\t\tError occurred.... recovering... \n",input+index);
186               pop();
187            }
188            else
189            {
190               display();
191               printf("\t\t%s\t\tREJECT\n",input+index);
192               break;
193            }
194         }
195      }
196   }
197 }
198
199
200
201 /***************INPUT**************************
202
203   5
204   E=(E)
205   E=E+E
206   E=E*E
207   E=a
208   a++aaa+
209
210 ***************************************************/
211
212
213 /*****************OUTPUT***********************
214
215   STACK     INPUT BUFFER     ACTION
216
217   $       a++aaa+$       SHIFT a
218   $a        ++aaa+$        REDUCE BY E=a
219   $E        ++aaa+$        SHIFT +
220   $E+         +aaa+$       Error.....operand missing.....recovering..
221   $E+         aaa+$        SHIFT a
222   $E+a      aa+$         Error.....operator missing.....recovering..
223   $E+a      a+$            Error.....operator missing.....recovering..
224   $E+a      +$           REDUCE BY E=a
225   $E+E      +$           REDUCE BY E=E+E
226   $E        +$           SHIFT +
227   $E+          $           Error occurred.... recovering...
228   $E         $           ACCEPT
229
230 ***************************************************/
```