```c
/*
 * ************************************************
 * ASSIGNMENT 8: LL(1) PARSER                     *
 * AUTHORS: SOUMYADIP MITRA AND SAIKAT KUMAR DEY  *
 * ROLL: 12/CS/39 AND 12/CS/40                    *
 *                                                *
 ************************************************
 **/


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

int top=1,i=0,l=0;
int flg=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')
```

```
58          return 2;
59      else if(a=='V')
60          return 3;
61      else if(a=='F')
62          return 4;
63      else if(a=='i')
64          return 5;
65      else if(a=='+')
66          return 6;
67      else if(a=='*')
68          return 7;
69      else if(a=='(')
70          return 8;
71      else if(a==')')
72          return 9;
73      else if(a=='$')
74          return 10;
75  }
76  int get_inp(char a)
77  {
78      if(a=='i')
79          return 0;
80      else if(a=='+')
81          return 1;
82      else if(a=='*')
83          return 2;
84      else if(a=='(')
85          return 3;
86      else if(a==')')
87          return 4;
88      else if(a=='$')
89          return 5;
90  }
91  void outputS()
92  {
93      int k;
94      printf("\n");
95      for(k=0;k<=top;k++)
96          printf("%c",stack[k]);
97  }
98  void outputI()
99  {
100     int k;
101     printf("\t\t");
102     for(k=i;k<l;k++)
103         printf("%c",inp[k]);
104 }
105 void parse()
106 {
107     int k,f=0;
108     printf("\nSTACK\t\tINPUT\t\tMESSAAGE");
109     char X,a;
110     outputS();
111     outputI();
112     while(1)
113     {
114         X=stack[top];
```

```
115             a=inp[i];
116             if(strcmp(table[get_stack(X)][get_inp(a)],"e1")==0)
117             {
118                 f=1;
119                 flg=0;
120                 printf("\t\tMISSING OPERAND : add 'i' onto input");
121                 for(k=l;k>=i;k--)
122                     inp[k+1]=inp[k];
123                 inp[k+1]='i';
124                 l++;
125                 outputS();
126                 outputI();
127             }
128             else if(X=='$' && a=='$')
129             {
130                 if(flg) printf("\nACCEPT");
131                 else
132                     printf("\nNOT ACCEPTED\n");
133                 if(f==1)
134                     printf("\nRECOVERED STRING : %s\n",inp);
135                 return;
136             }
137             else if(X=='$')
138             {
139                 f=1;
140                 printf("\t\tUNEXPECTED %c ",a);
141                 inp[i]='$';
142                 inp[i+1]='\0';
143                 l=i+1;
144             }
145             else if(X==')'&& a!=')')
146             {
147                 f=1;
148                 printf("\t\tMISSING RIGHT PARENTHESIS");
149                 for(k=l;k>=i;k--)
150                     inp[k+1]=inp[k];
151                 inp[k+1]=')';
152                 l++;
153                 outputS();
154                 outputI();
155             }
156             else if(X==a)
157             {
158                 pop();
159                 i++;
160                 outputS();
161                 outputI();
162             }
163             else
164             {
165                 pop();
166                 strcpy(t,table[get_stack(X)][get_inp(a)]);
167                 rev(t);
168                 if(strcmp(t,"#")!=0)
169                 {
170                     strcat(stack,t);
171                     top=top+strlen(t);
```

```
172             }
173             outputS();
174             outputI();
175             printf("\t\t%c->%s",X,table[get_stack(X)][get_inp(a)]);
176         }
177     }
178 }
179
180 int main()
181 {
182     printf("GRAMMER :");
183     printf("\nE->TU");
184     printf("\nU->+TU|#");
185     printf("\nT->FV");
186     printf("\nV->*FV|#");
187     printf("\nF->(E)|i");
188     printf("\n\nwhere U stands for E', V stands for T', i stands for Id and #
    stands for NULL\n");
189     printf("\nEnter the string to be parsed\n\n");
190     gets(inp);
191     printf("\n\n");
192     l=strlen(inp);
193     inp[l]='$';
194     inp[l+1]='\0';
195     l++;
196     parse();
197     return 0;
198 }
199
200 /*
201                 *****OUTPUT****************
202
203 GRAMMER :
204 E->TU
205 U->+TU|#
206 T->FV
207 V->*FV|#
208 F->(E)|i
209
210 where U stands for E', V stands for T', i stands for Id and # stands for NULL
211
212 Enter the string to be parsed
213
214 i+i*i
215
216 STACK       INPUT       MESSAAGE
217 $E          i+i*i$
218 $UT         i+i*i$      E->TU
219 $UVF        i+i*i$      T->FV
220 $UVi        i+i*i$      F->i
221 $UV         +i*i$
222 $U          +i*i$       V->#
223 $UT+        +i*i$       U->+TU
224 $UT         i*i$
225 $UVF        i*i$        T->FV
226 $UVi        i*i$        F->i
227 $UV         *i$
```

```
$UVF*       *i$         V->*FV
$UVF        i$
$UVi        i$          F->i
$UV         $
$U          $           V->#
$           $           U->#

            ACCEPT


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+

STACK       INPUT       MESSAAGE
$E          i+i*i+$
$UT         i+i*i+$     E->TU
$UVF        i+i*i+$     T->FV
$UVi        i+i*i+$     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
$UV         *i+$
$UVF*       *i+$        V->*FV
$UVF        i+$
$UVi        i+$     F->i
$UV         +$
$U          +$      V->#
$UT+        +$      U->+TU
$UT         $       MISSING OPERAND : add 'i' onto input
$UT         i$
$UVF        i$      T->FV
$UVi        i$      F->i
$UV         $
$U          $       V->#
$           $       U->#


    PARSED STRING  NOT ACCEPTED


*/
```