

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  int top=1,i=0,l=0;
6  int flag=1;
7  char stack[100]="$E",inp[100],t[100];
8  char table[11][6][10]={{"TU","e1","e1","TU","e1","e1"},
9      {"e1","+TU","e1","e1","#", "#"},
10     {"FV","e1","e1","FV","e1","e1"},
11     {"e1","#","*FV","e1","#", "#"},
12     {"i","e1","e1","(E)","e1","e1"},
13     {"pop"," "," "," "," "," "},
14     {" ","pop"," "," "," "," "},
15     {" "," ","pop"," "," "," "},
16     {" "," "," ","pop"," "," "},
17     {"e2","e2","e2","e2","pop","e2"},
18     {"e3","e3","e3","e3","e3","accept"}};
19
20 void pop()
21 {
22     stack[top--]='\0';
23 }
24 int terminal(char a)
25 {
26     if(a=='i' || a=='+' || a=='*' || a=='(' || a=='(' || a=='$')
27         return 1;
28     else
29         return 0;
30 }
31 void rev(char *s)
32 {
33     int j;
34     char b;
35     for(j=0;j<strlen(s)/2;j++){
36         b=s[j];
37         s[j]=s[strlen(s)-j-1];
38         s[strlen(s)-j-1]=b;
39     }
40 }
41 int get_stack(char a)
42 {
43     if(a=='E')
44         return 0;
45     else if(a=='U')
46         return 1;
47     else if(a=='T')
48         return 2;
49     else if(a=='V')
50         return 3;
51     else if(a=='F')
52         return 4;
53     else if(a=='i')
54         return 5;
55     else if(a=='+')
56         return 6;
57     else if(a=='*')
58         return 7;
59     else if(a=='(')
60         return 8;
61     else if(a==')')
62         return 9;
63     else if(a=='$')
64         return 10;
65 }
66 int get_inp(char a)
67 {
68     if(a=='i')
69         return 0;
70     else if(a=='+')
71         return 1;
72     else if(a=='*')
73         return 2;
74     else if(a=='(')
75         return 3;
76     else if(a==')')
77         return 4;

```

```

78     else if(a=='$')
79         return 5;
80 }
81 void outputS()
82 {
83     int k;
84     printf("\n");
85     for(k=0;k<=top;k++)
86         printf("%c",stack[k]);
87 }
88 void outputI()
89 {
90     int k;
91     printf("\t\t");
92     for(k=i;k<=l;k++)
93         printf("%c",inp[k]);
94 }
95 void parse()
96 {
97     int k,f=0;
98     printf("\nSTACK\t\tINPUT\t\tMESSAGE");
99     char X,a;
100    outputS();
101    outputI();
102    while(1)
103    {
104        X=stack[top];
105        a=inp[i];
106        if(strcmp(table[get_stack(X)][get_inp(a)],"e1")==0)
107        {
108            f=1;
109            printf("\t\tMISSING OPERAND : add 'i' onto input");
110            flag=0;
111            for(k=l;k>=i;k--)
112                inp[k+1]=inp[k];
113            inp[k+1]='i';
114            l++;
115            outputS();
116            outputI();
117        }
118        else if(X=='$' && a=='$')
119        {
120            if(flag) printf("\nACCEPT");
121            else printf("\nPARSED STRING NOT ACCEPTED");
122            if(f==1)
123                printf("\nRECOVERED STRING : %s",inp);
124            return;
125        }
126        else if(X=='$')
127        {
128            f=1;
129            printf("\t\tUNEXPECTED %c ",a);
130            inp[i]='$';
131            inp[i+1]='\0';
132            l=i+1;
133        }
134        else if(X==')' && a!=')')
135        {
136            f=1;
137            printf("\t\tMISSING RIGHT PARENTHESIS");
138            for(k=l;k>=i;k--)
139                inp[k+1]=inp[k];
140            inp[k+1]=')';
141            l++;
142            outputS();
143            outputI();
144        }
145        else if(X==a)
146        {
147            pop();
148            i++;
149            outputS();
150            outputI();
151        }
152        else
153        {
154            pop();

```

```

155     strcpy(t,table[get_stack(X)][get_inp(a)]);
156     rev(t);
157     if(strcmp(t,"")!=0)
158     {
159         strcat(stack,t);
160         top=top+strlen(t);
161     }
162     outputS();
163     outputI();
164     printf("\t\t%c->%s",X,table[get_stack(X)][get_inp(a)]);
165     }
166 }
167 }
168
169 int main()
170 {
171     printf("GRAMMER :");
172     printf("\nE->TU");
173     printf("\nU->+TU|#");
174     printf("\nT->FV");
175     printf("\nV->*FV|#");
176     printf("\nF->(E)|i");
177     printf("\n\nwhere U stands for E', V stands for T', i stands for Id and # stands for NULL\n");
178     printf("\nEnter the string to be parsed\n\n");
179     gets(inp);
180     printf("\n\n");
181     l=strlen(inp);
182     inp[l]='$';
183     inp[l+1]='\0';
184     l++;
185     parse();
186     return 0;
187 }
188
189
190 /*
191 GRAMMER :
192 E->TU
193 U->+TU|#
194 T->FV
195 V->*FV|#
196 F->(E)|i
197
198 where U stands for E', V stands for T', i stands for Id and # stands for NULL
199
200 Enter the string to be parsed
201 i+i*i+i+i
202
203 STACK    INPUT    MESSAAGE
204 $E      i+i*i+i+i$
205 $UT     i+i*i+i+i$  E->TU
206 $UVF    i+i*i+i+i$  T->FV
207 $UVi    i+i*i+i+i$  F->i
208 $UV     +i*i+i+i$
209 $U      +i*i+i+i$   V->#
210 $UT+    +i*i+i+i$   U->+TU
211 $UT     i*i+i+i$
212 $UVF    i*i+i+i$   T->FV
213 $UVi    i*i+i+i$   F->i
214 $UV     *i+i+i$
215 $UVF*    *i+i+i$   V->*FV
216 $UVF    i+i+i$
217 $UVi    i+i+i$   F->i
218 $UV     +i+i$
219 $U      +i+i$   V->#
220 $UT+    +i+i$   U->+TU
221 $UT     i+i$
222 $UVF    i+i$   T->FV
223 $UVi    i+i$   F->i
224 $UV     +i$
225 $U      +i$   V->#
226 $UT+    +i$   U->+TU
227 $UT     i$
228 $UVF    i$   T->FV
229 $UVi    i$   F->i
230 $UV     $
231 $U      $   V->#

```

232 \$ \$ U->#

233 ACCEPT

234 */

235