```
/*
 * AUTHORS: BIJAY KHATRI & DEEPAK VERMA
 * ROLL: 12/CS/45 AND 12/CS/46
 */

#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=='*')
```

```
78        return 2;
79     else if(a=='(')
80        return 3;
81     else if(a==')')
82        return 4;
83     else if(a=='$')
84        return 5;
85  }
86  void outputS()
87  {
88     int k;
89     printf("\n");
90     for(k=0;k<=top;k++)
91        printf("%c",stack[k]);
92  }
93  void outputI()
94  {
95     int k;
96     printf("\t\t");
97     for(k=i;k<l;k++)
98        printf("%c",inp[k]);
99  }
100 void parse()
101 {
102    int k,f=0;
103    printf("\nSTACK\t\tINPUT\t\tMESSAAGE");
104    char X,a;
105    outputS();
106    outputI();
107    while(1)
108    {
109       X=stack[top];
110       a=inp[i];
111       if(strcmp(table[get_stack(X)][get_inp(a)],"e1")==0)
112       {
113          f=1;
114          printf("\t\tMISSING OPERAND : add 'i' onto input");
115          flag=0;
116          for(k=l;k>=i;k--)
117             inp[k+1]=inp[k];
118          inp[k+1]='i';
119          l++;
120          outputS();
121          outputI();
122       }
123       else if(X=='$' && a=='$')
124       {
125         if(flag) printf("\nACCEPT");
126         else printf("\nPARSED STRING NOT ACCEPTED");
127          if(f==1)
128             printf("\nRECOVERED STRING : %s",inp);
129          return;
130       }
131       else if(X=='$')
132       {
133          f=1;
134          printf("\t\tUNEXPECTED %c ",a);
135          inp[i]='$';
136          inp[i+1]='\0';
137          l=i+1;
138       }
139       else if(X==')'&& a!=')')
140       {
141          f=1;
142          printf("\t\tMISSING RIGHT PARENTHESIS");
143          for(k=l;k>=i;k--)
144             inp[k+1]=inp[k];
145          inp[k+1]=')';
146          l++;
147          outputS();
148          outputI();
149       }
150       else if(X==a)
151       {
152          pop();
153          i++;
154          outputS();
```

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

```
$UT     i$
$UVF    i$      T->FV
$UVi    i$      F->i
$UV     $
$U    $   V->#
$   $   U->#
ACCEPT
*/

```