```c
/*
                                Assignment 6

            Implement an operator precedence parser assuming conventional precedence of operator
            for construction of precedence table. Your parser would be able to perform error recovery
            and detection and print diagnostic message.

Compiled By:
Himansu Rathi  12/CS/06
R Om prakash   12/CS/07

*/
#include<stdio.h>
#include<ctype.h>
#include<string.h>

struct stru1
{
char non_ter[1],pro[25];
}cfg[25];
int n,st=-1,j,i,t=-1,m,in=0;
int v,c,p=1,s=0;
char str[20],stack[20],ch,tmp[10],input[20],symbol[2];
int main()
{
char temp[20];
char prec[6][6]={{' ','-','+','*','/','i'},{'-','=','<','<','<','<'},{'+','>','=',
'<','<','<'},{'*','>','>','=','<','<'},{'/','>','>','>','=','<'},{'i','>','>','>',
'>','='}};
printf("\n\n");
for(i=0;i<6;i++)
{
    for(j=0;j<6;j++)
        printf("  %c |",prec[i][j]);
    printf("\n---------------------------------\n");
}
printf("Enter the number of productions:\n\r");
scanf("%d",&n);
printf("\n\r");
printf("Enter the productions:\n\r");
for(i=0;i<n;i++)
{
scanf("%s",cfg[i].non_ter);
//printf("\n\r");
scanf("%s",symbol);
scanf("%s",cfg[i].pro);
printf("\n\r");
}
printf("Enter the input string:\n\r");
scanf("%s",str);
printf("\n\r");
int i,r;
printf("ACTION\t\tSTACK\t\tINPUT\n");
printf("-\t\t-\t\t%s$\n",str);
label1: stack[++st]=str[s++];
```

```
54  int low=0;
55          for(j=s;j<strlen(str);j++)
56          {
57              input[low++]=str[j];
58          }m++;
59          input[low++]='$';
60          input[low]='\0';
61  label3:     if(str[s-1]=='\0')
62          {
63
64              if(stack[st-1]==')')
65              {
66                  printf("unbalanced right paranthesis \n Recovering error : removed
                    right paranthesis\n");
67                  if(st==2)
68                  {
69                  stack[st-1]='\0';
70                  goto label3;
71                  }
72                  st--;
73                  printf("removed ')'\t%s\t%s\n",stack,input);
74                  goto label;
75              }
76              else if(st==1)
77              printf("\n\n\tString accepted\n");
78              else
79              printf("\n\n\tNot accepted by above grammar\n");
80              return 0;
81          }
82          printf("shift %c\t\t%s\t\t%s\n",stack[st],stack,input);
83          if((stack[st]=='+'||stack[st]=='-'||stack[st]=='*'||stack[st]=='/')&&(input
            [0]=='+'||input[0]=='-'||input[0]=='*'||input[0]=='/'))
84              printf("\n---MISSING OPERAND---\n");
85      //  scanf("%d",&t);
86  label: for (i=0;i<=st;i++)
87  {
88      int l=0;
89      for (j=i;j<=st;j++)
90      {
91          temp[l++]=stack[j];
92      }
93      temp[l]='\0';
94      for (r=0;r<n;r++){
95      if(strcmp(temp,cfg[r].pro)==0)
96      {
97          int len = strlen(temp);
98          while(len--)
99          {
100             stack[st--]=NULL;
101         }
102         stack[++st]=cfg[r].non_ter[0];
103         printf ("Reduced %c->%s\t%s\t\t%s\n",cfg[r].non_ter[0],cfg[r].pro,stack,
            input);
104
105         if((stack[st-1]=='+'||stack[st-1]=='-'||stack[st-1]=='*')&&input[0]=='/')
106             goto label1;
107         else if((stack[st-1]=='+'||stack[st-1]=='-')&&input[0]=='*')
```

```
108                 goto label1;
109             else if((stack[st-1]=='-')&&input[0]=='+')
110                 goto label1;
111
112             if((isalnum(stack[st]))&&isalnum(stack[st-1]))
113                 printf("\n---MISSING OPERATOR---\n");
114             goto label;
115         }
116         }
117     }
118     goto label1;
119     return 0;
120     }
121
```