1. Two Pass Assembler

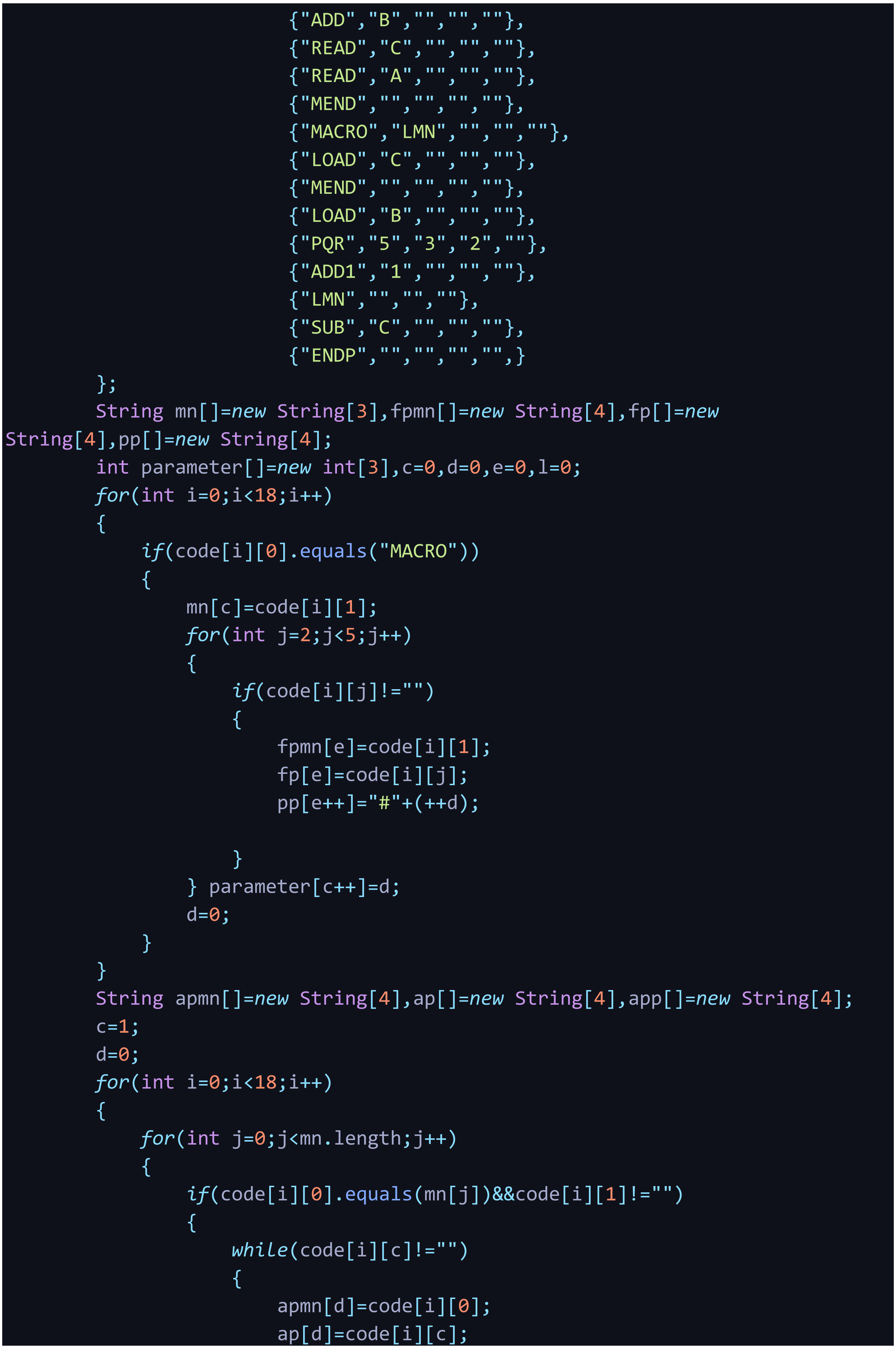
```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
    char *code[9][4]={
            {"PRG1","START","",""},
            {"","USING","*","15"},
            {"","L","",""},
            {"","A","",""},
            {"","ST","",""},
            {"FOUR","DC","F",""},
            {"FIVE","DC","F",""},
            {"TEMP","DS","F",""},
            {"","END","",""}
    };
    char
av[2],avail[15]={'N','N','N','N','N','N','N','N','N','N','N','N','N','N','N
'};
    int i,j,k,count[3],lc[9]={0,0,0,0,0,0,0,0,0},loc=0;
    clrscr();
    printf("------------------------------\n");
    printf("LABLE\t\t\OPCODE\n");
    printf("------------------------------\n\n");
    for(i=0;i<=8;i++)
    {
        for(j=0;j<=3;j++)
        {
            printf("%s\t\t",code[i][j]);
        }
        printf("\n");
    }
    getch();
    printf("------------------------");
    printf("\n VALUES FOR LC:\n\n");
    for(j=0;j<=8;j++)
    {

if((strcmp(code[j][1],"START")!=0)&&(strcmp(code[j][1],"USING")!=0)&&(strcm
p(code[j][1],"L")!=0)
                )
            lc[j]=lc[j-1]+4;
        printf("%d\t",lc[j]);
    }
    printf("\n\nSYMBOL TABLE:\n-----------------------------\n");
    printf("SYMBOL\t\tVALUE\t\tLENGTH\t\tR/A");
    printf("\n-------------------------\n");
    for(i=0;i<9;i++)
    {
        if(strcmp(code[i][1],"START")==0)
        {
            printf("%s\t\t%d\t\t%d\t\t%c\n",code[i][0],loc,4,'R');
        }
        else if(strcmp(code[i][0],"")!=0)

        {
            printf("%s\t\t%d\t\t%d\t\t%c\n",code[i][0],loc,4,'R');
            loc=loc+4;
        }
        else if(strcmp(code[i][1],"USING")==0){}
        else
        {loc=loc+4;}
    }
    printf("-----------------------------------");
    printf("\n\nBASE TABLE:\n-----------------------------\n");
    printf("REG NO\t\tAVAILIBILITY\tCONTENTS OF BASE TABLE");
    printf("\n------------------------------------\n");
    for(j=0;j<=8;j++)
    {
        if(strcmp(code[j][1],"USING")!=0)
        {}
        else
        {
            strcpy(av,code[j][3]);
        }
    }
    count[0]=(int)av[0]-48;
    count[1]=(int)av[1]-48;
    count[2]=count[0]*10+count[1];
    avail[count[2]-1]='Y';
    for(k=0;k<16;k++)
    {
        printf("%d\t\t%c\n",k,avail[k-1]);
    }
    printf("----------------------------------\n");
    printf("Continue..??\n\n");
    getch();
    printf("PASS2 TABLE:\n\n");
    printf("LABLE\t\tOP1\t\tLC\t\t");
    printf("\n---------------------\n");
    loc=0;
    for(i=0;i<=8;i++)
    {
        for(j=0;j<=3;j++)
        {
            printf("%s\t\t",code[i][j]);
        }
        j=0;
        printf("\n");
    }
    printf("------------------------------");
    getch();
}
```

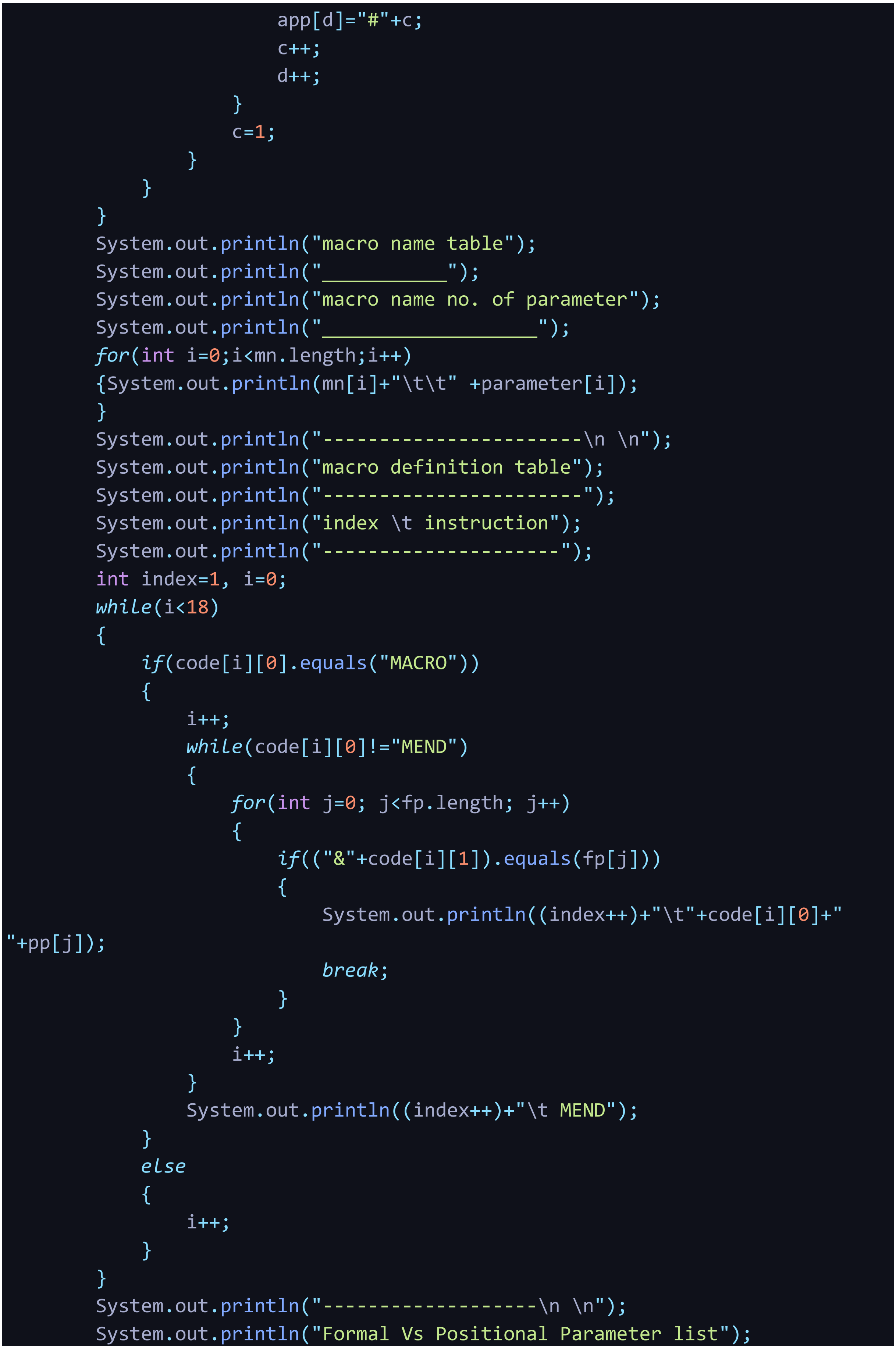
2. To implement a two pass macro pre-processor

```
import java.io.*;
class Macroprocessor
{
public static void main(String args[])
    {
        String code[][]={{"ADD","A","","",""},
                         {"MACRO","ADD1","&ARG","",""},
                         {"LOAD","ARG","","",""},
                         {"MEND","","","",""},
                         {"MACRO","PQR","&A","&B","&C"},
```

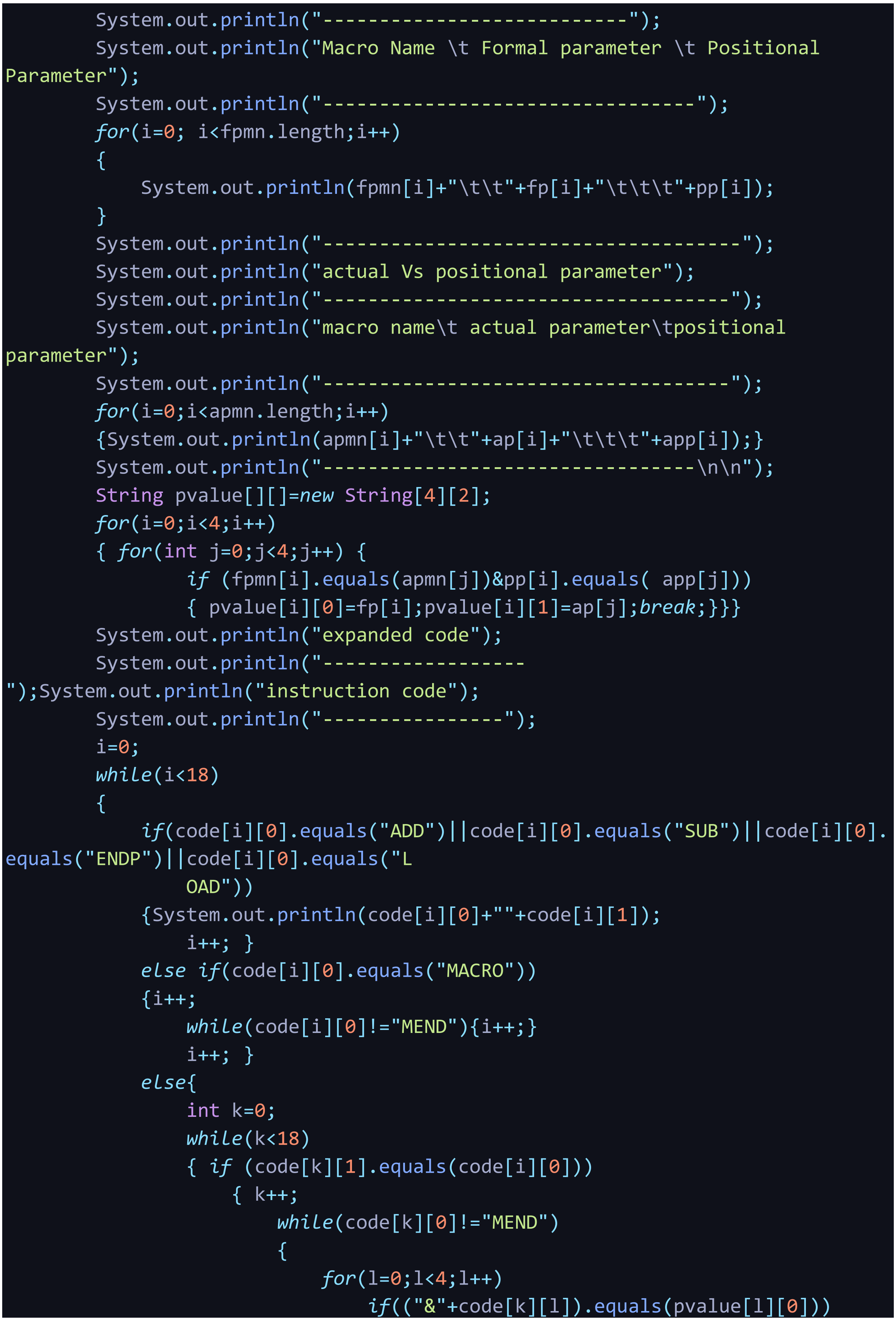


```
            {"ADD","B","","",""},
            {"READ","C","","",""},
            {"READ","A","","",""},
            {"MEND","","","",""},
            {"MACRO","LMN","","",""},
            {"LOAD","C","","",""},
            {"MEND","","","",""},
            {"LOAD","B","","",""},
            {"PQR","5","3","2",""},
            {"ADD1","1","","",""},
            {"LMN","","",""},
            {"SUB","C","","",""},
            {"ENDP","","","","",}
    };
    String mn[]=new String[3],fpmn[]=new String[4],fp[]=new
String[4],pp[]=new String[4];
    int parameter[]=new int[3],c=0,d=0,e=0,l=0;
    for(int i=0;i<18;i++)
    {
        if(code[i][0].equals("MACRO"))
        {
            mn[c]=code[i][1];
            for(int j=2;j<5;j++)
            {
                if(code[i][j]!="")
                {
                    fpmn[e]=code[i][1];
                    fp[e]=code[i][j];
                    pp[e++]="#"+(++d);

                }
            } parameter[c++]=d;
            d=0;
        }
    }
    String apmn[]=new String[4],ap[]=new String[4],app[]=new String[4];
    c=1;
    d=0;
    for(int i=0;i<18;i++)
    {
        for(int j=0;j<mn.length;j++)
        {
            if(code[i][0].equals(mn[j])&&code[i][1]!="")
            {
                while(code[i][c]!="")
                {
                    apmn[d]=code[i][0];
                    ap[d]=code[i][c];
```



```
                        app[d]="#"+c;
                        c++;
                        d++;
                    }
                    c=1;
                }
            }
        }
        System.out.println("macro name table");
        System.out.println("___________");
        System.out.println("macro name no. of parameter");
        System.out.println("___________________");
        for(int i=0;i<mn.length;i++)
        {System.out.println(mn[i]+"\t\t" +parameter[i]);
        }
        System.out.println("-----------------------\n \n");
        System.out.println("macro definition table");
        System.out.println("-----------------------");
        System.out.println("index \t instruction");
        System.out.println("---------------------");
        int index=1, i=0;
        while(i<18)
        {
            if(code[i][0].equals("MACRO"))
            {
                i++;
                while(code[i][0]!="MEND")
                {
                    for(int j=0; j<fp.length; j++)
                    {
                        if(("&"+code[i][1]).equals(fp[j]))
                        {
                            System.out.println((index++)+"\t"+code[i][0]+"
"+pp[j]);
                            break;
                        }
                    }
                    i++;
                }
                System.out.println((index++)+"\t MEND");
            }
            else
            {
                i++;
            }
        }
        System.out.println("-------------------\n \n");
        System.out.println("Formal Vs Positional Parameter list");
```



```
        System.out.println("---------------------------");
        System.out.println("Macro Name \t Formal parameter \t Positional 
Parameter");
        System.out.println("---------------------------------");
        for(i=0; i<fpmn.length;i++)
        {
            System.out.println(fpmn[i]+"\t\t"+fp[i]+"\t\t\t"+pp[i]);
        }
        System.out.println("-------------------------------------");
        System.out.println("actual Vs positional parameter");
        System.out.println("------------------------------------");
        System.out.println("macro name\t actual parameter\tpositional 
parameter");
        System.out.println("------------------------------------");
        for(i=0;i<apmn.length;i++)
        {System.out.println(apmn[i]+"\t\t"+ap[i]+"\t\t\t"+app[i]);}
        System.out.println("---------------------------------\n\n");
        String pvalue[][]=new String[4][2];
        for(i=0;i<4;i++)
        { for(int j=0;j<4;j++) {
                if (fpmn[i].equals(apmn[j])&pp[i].equals( app[j]))
                { pvalue[i][0]=fp[i];pvalue[i][1]=ap[j];break;}}}
        System.out.println("expanded code");
        System.out.println("------------------
");System.out.println("instruction code");
        System.out.println("----------------");
        i=0;
        while(i<18)
        {
            if(code[i][0].equals("ADD")||code[i][0].equals("SUB")||code[i][0].
equals("ENDP")||code[i][0].equals("L
                OAD"))
            {System.out.println(code[i][0]+""+code[i][1]);
                i++; }
            else if(code[i][0].equals("MACRO"))
            {i++;
                while(code[i][0]!="MEND"){i++;}
                i++; }
            else{
                int k=0;
                while(k<18)
                { if (code[k][1].equals(code[i][0]))
                    { k++;
                        while(code[k][0]!="MEND")
                        {
                            for(l=0;l<4;l++)
                                if(("&"+code[k][l]).equals(pvalue[l][0]))
```

```
                                    System.out.println(code[k][0]+""+pvalue[l]
[l]);
                        }
                        k++; }k++; }k++; i++; }
        } } }
```

3. To design a lexical analyzer for a language whose grammar is known.

```
LEX Program <x1,e>

%{
#include"y.tab.h"
extern int yylval;
%}
%%
[0-9]+ {yylval=ato(yytext);
return NUM;
}
return yytext[0];
\n return 0;
%%
int yywrap();
{
return 1;
}
```

```
YACC Program <x1,y>
%{
#include<stdio.h>
%}
%token A NUM
%%
state: A'='E
|E {print("\n The result=%d\n",$1);}
;
E:E'+NUM {$S=$1+$3;}
|NUM {$S=$1;}
;
%%
extern FILE *yyin;
main()
{
    do
    {
        yyparse();
    }while(!feof(yyin));
}
yyerror(char *s)
{
    fprintf(stderr,"%s\n",s);
}
```

OUTPUT:
[root@aap root]# lex x1.1
[root@aap root]# yacc -d x1.y

```
[root@aap root]# cc lex.yy.c.y.tab.c
[root@aap root]# ./a.out 4+6 The result=10
```

```
LEX Program <anbn.I>
%{
#include"y.tab.h"
%}
%%
a {return A;}
b {return B;}
. {return(yytext[0]);}
\n return ('\n');
%%
int yywrap()
{
    return 1;
}
```

```

YACC Program <anbn.y>
%{
%}
%token A B
%%
statement:anbn'\n' {printf("\n Its a valid string!!!");
return 0;}
anbn: A B
|A anbn B
;
%%
main();
{
printf("\n Enter some valid string\n");
yyparse();
}
int yyerror(char *s)
{
    printf("\nIt is not in anbn");
}
```

OUTPUT (run 1):

```
[root@aap root]# lex x2.1
[root@aap root]# yacc x2.y
[root@aap root]# cc lex.yy.c.y.tab.c
[root@aap root]# ./a.out
Enter some valid string
aabb
Its a valid string!!!
```

OUTPUT (run 2):

```
[root@aap root]# ./a.out
Enter some valid string
abbb
It is not in anbn

OUTPUT (run 3):
[root@aap root]# ./a.out
Enter some valid string
It is not in anbn
```

4. To implement a simple parser using Lex YACC

```
//lex prgram
%{
#include “y.tab.h”
Extern int yylval
%}
%%
[0-9]+{yylval=atoi(yytext);
Return NUM;
}
Return yytext[0];
\n return 0;
%%
Int yywrap()
{
    Return1;
}
```

```
// yacc program<x1.y>
%{
Include<stdio.h>
%}
% token A NUM
%%
State: A’=’E | E {printf(“\n the result=%d\N”,$1);}
;
E:E’+’Nunm{$$=$1+$3;}
| NUM { $$=$1;}
;
%%
Extern FILE yyin;
Main()
{
    Do
    {
        Yyparse();
    }
    While(!feof(yyin))
}
Yyerror(char *s)
{
    Fprintf(stderr,”%s\n”,s);
}
```

Output

```
[root@app root]$ lex x1.l
[root@app root]$ yacc –d x1.y
[root@app root]$ cc lex.yy.c y.tab.c
[root@app root]$ ./a.out

4+5

The result=10
```

5. To perform code optimization, considering the target machine to be X86

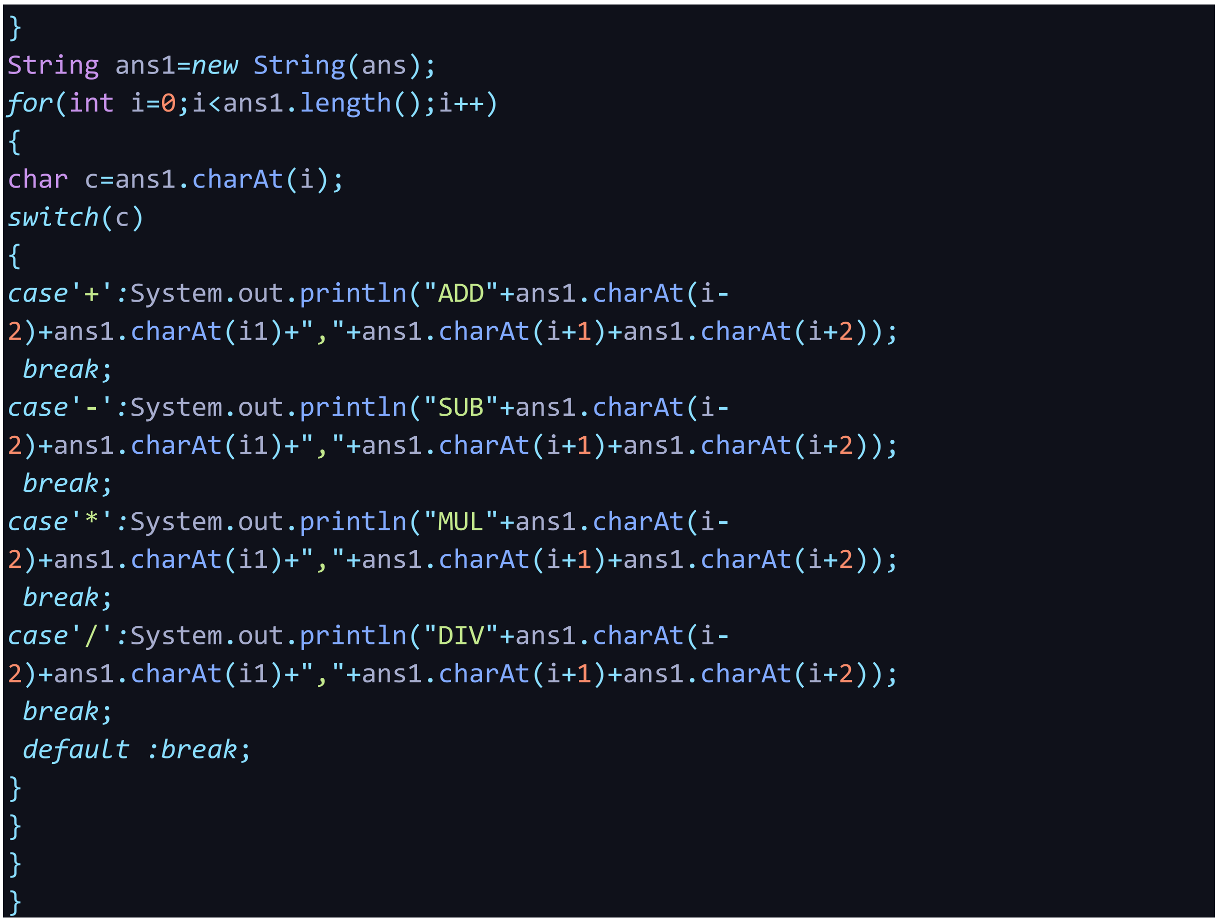
```
Import java.ip.*;
Import java.util.*;
Import java.long.String;
Public class optimization
{
Public static void main(String args[]) throws IOException
{
DataInputStream in=new DataInputStream(System.in);
String s1,s2;
String code[]=new String[10];
System.out.println(“Enter the string1:);
S1=in.readLine();
System.out.println(“Enter the string2:);
S2=in.readLine();
If(s1.equals(s2))
{
System.out.println(“enter string is duplicate”);
S2=null;
}
Else
System.out.println(“enter string is correct”);
System.out.println(“enter the line of code”);
Int n=Integer.praseInt(in.readLine());
System.out.println(“enter the code of program”);
For(int i=0;i<=n;i++)
Code[i]=in.readLine();
For(int i=0;i<n;i++)
{
Char c[]=code[i].toCharArray();
Char d[]=code[i+1].toCharArray();
If ((c[0]==’I’)&&(c[1]==’n’)&&(c[2]==’t’))
If(d[3]==c[4])
System.out.println(“the line”+code[i+1]+”will not be excuted since it’s a dead
code”)’
Else
System.out.println(“code is corrected”);
}}}
```

Output:
Enter the string1
Int i=0;
Enter the string2
Int j=3;
Enter string is corrected
Enter the line of code:
3
Enter the code of program
Int k=0; If (k) K=K+1;
The line if(k) will not be excuted since it’s a dead code

6. To generate target code for the code optimized, considering the target machine to be X86

```
import java.io.*;
public class exp6
{
 public static void main(String args[])throws IOException
{
DataInputStream in=new DataInputStream(System.in);
System.out.println("Enter the equation");
String stmt=in.readLine();
StringBuffer ans=new StringBuffer("");
int reg=0;
int count=0;
char c2='a';
int flag=0;
for(int i=0;i<stmt.length();i++)
{
char c=stmt.charAt(i);
if(i>0)
{
c2=stmt.charAt(i-1);
}
switch(c)
{
case'(':count++;
break;
case')':count--;
break;
case'+':if(count>0)
{
System.out.println("MOV "+stmt.charAt(i-1)+",R"+reg);
```

```java
System.out.println("ADD "+stmt.charAt(i+1)+",R"+reg);
ans.append("R"+reg);
reg++;
}
else
{
ans.append("+");
}
break;
case'-':if(count>0)
{
System.out.println("MOV "+stmt.charAt(i-1)+",R"+reg);
System.out.println("SUB "+stmt.charAt(i+1)+",R"+reg);
ans.append("R"+reg);
reg++;
}
else
{
ans.append("-");
}
break;
case'*':if(count>0)
{
System.out.println("MOV "+stmt.charAt(i-1)+",R"+reg);
System.out.println("MUL "+stmt.charAt(i+1)+",R"+reg);
ans.append("R"+reg);
reg++;
}
else
{
ans.append("*");
}
break;
case'/':if(count>0)
{
System.out.println("MOV "+stmt.charAt(i-1)+",R"+reg);
System.out.println("DIV "+stmt.charAt(i+1)+",R"+reg);
ans.append("R"+reg);
reg++;
}
else
{
ans.append("/");
}
break;
default:break;
}
flag++;
```

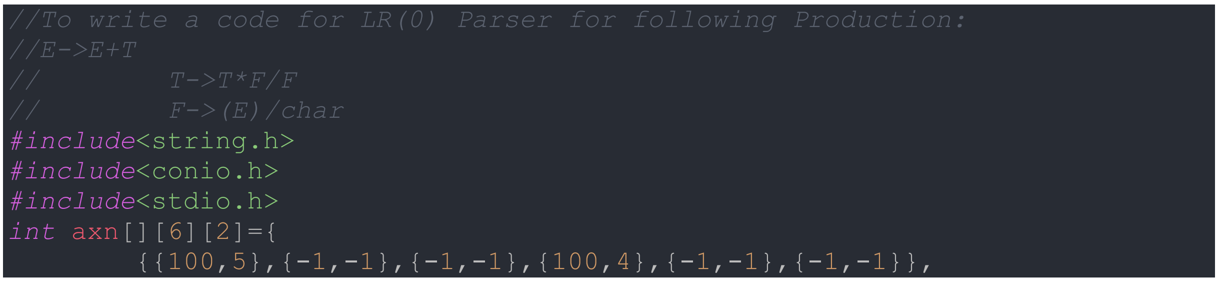


```
}
String ans1=new String(ans);
for(int i=0;i<ans1.length();i++)
{
char c=ans1.charAt(i);
switch(c)
{
case'+':System.out.println("ADD"+ans1.charAt(i-
2)+ans1.charAt(i1)+","+ans1.charAt(i+1)+ans1.charAt(i+2));
 break;
case'-':System.out.println("SUB"+ans1.charAt(i-
2)+ans1.charAt(i1)+","+ans1.charAt(i+1)+ans1.charAt(i+2));
 break;
case'*':System.out.println("MUL"+ans1.charAt(i-
2)+ans1.charAt(i1)+","+ans1.charAt(i+1)+ans1.charAt(i+2));
 break;
case'/':System.out.println("DIV"+ans1.charAt(i-
2)+ans1.charAt(i1)+","+ans1.charAt(i+1)+ans1.charAt(i+2));
 break;
 default :break;
}
}
}
}
```

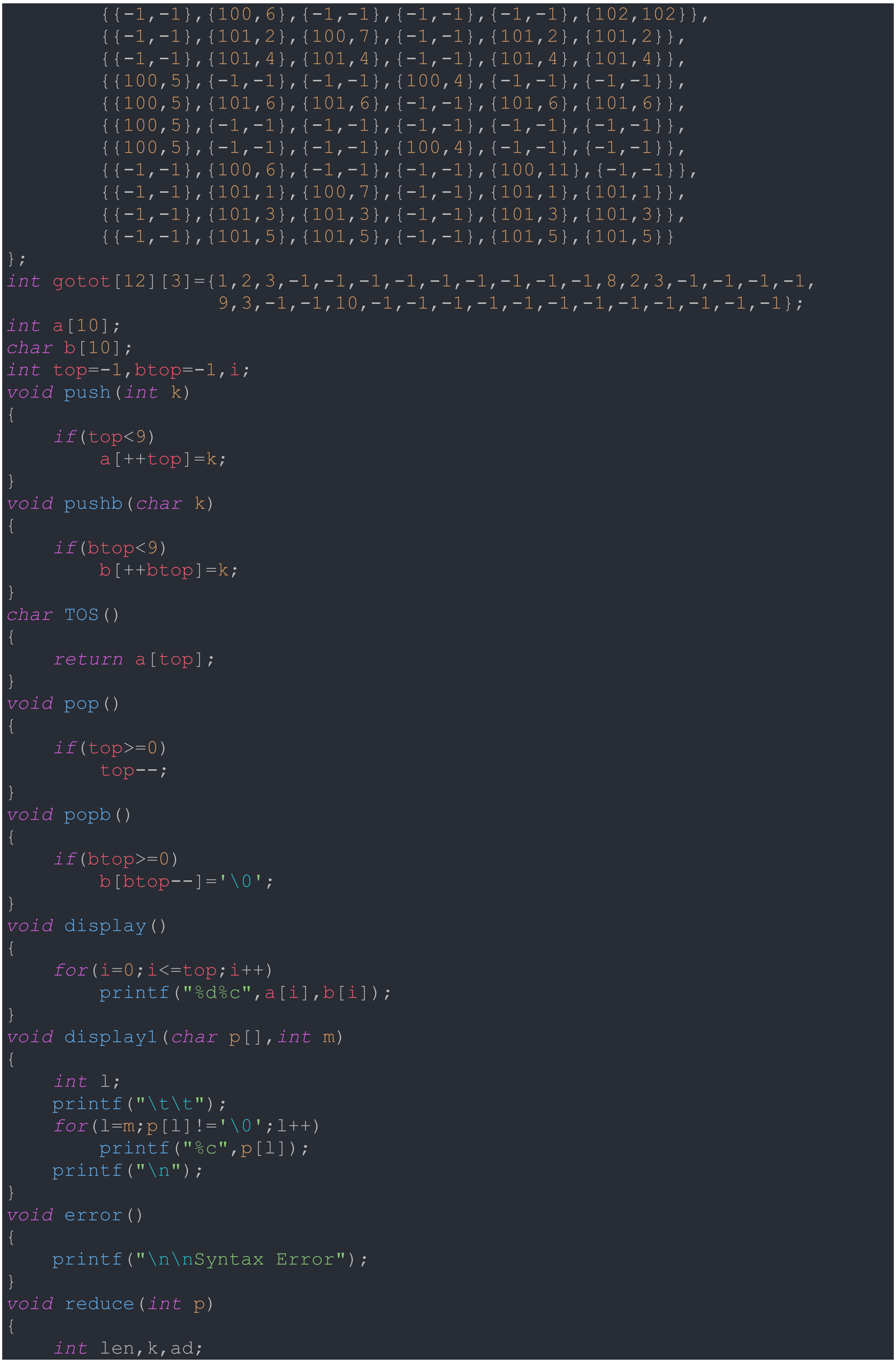
OUTPUT

```
Enter the equation
(a+b)*(c-d)+(e/f)*(a+b)
MOV a,R0
ADD b,R0
MOV c,R1
SUB d,R1
MOV e,R2
DIV f,R2
MOV a,R3
ADD b,R3
MUL R0,R1
ADD R1,R2
MUL R2,R3
```

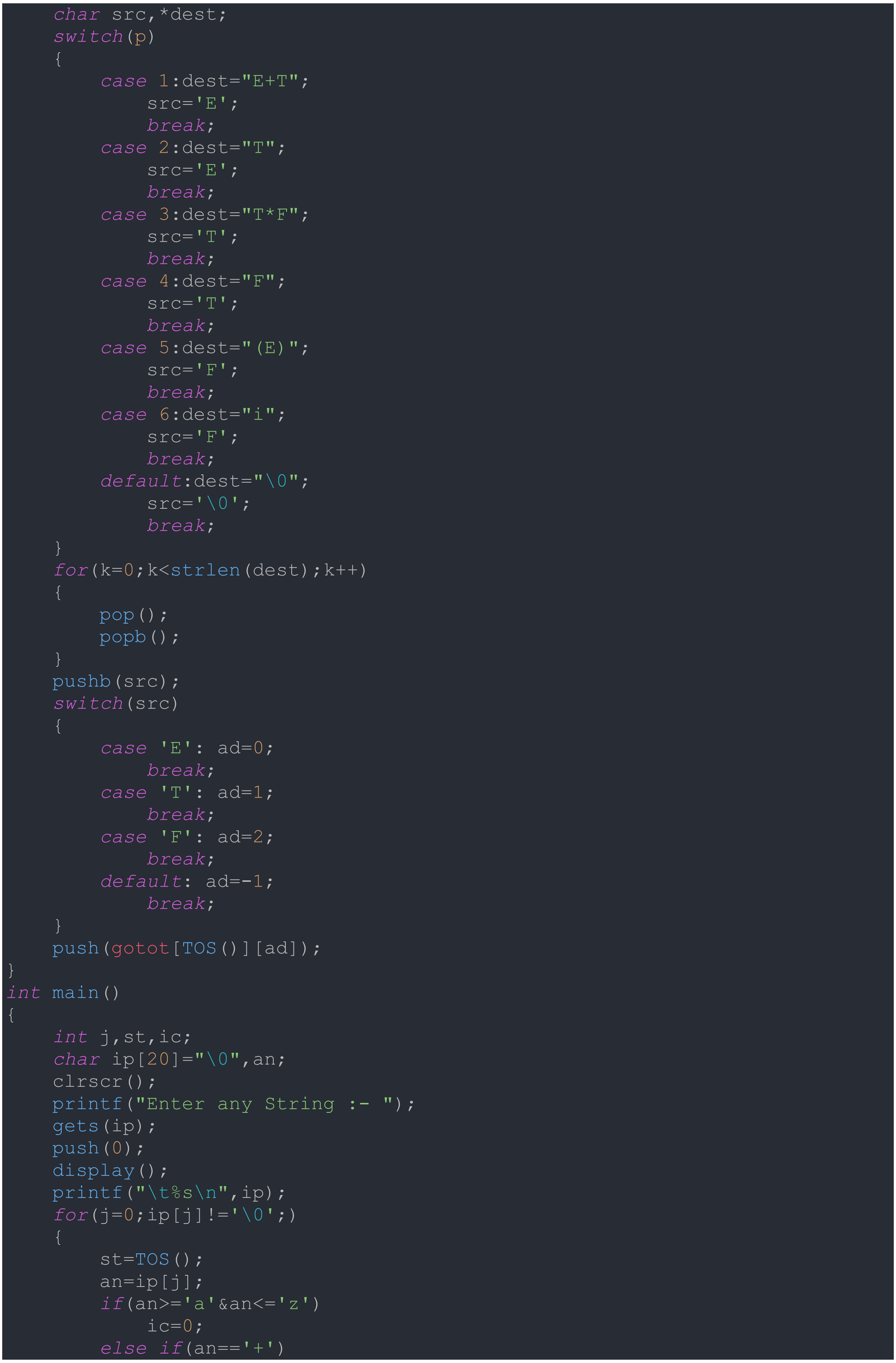
7. To implement a LR(0) parser



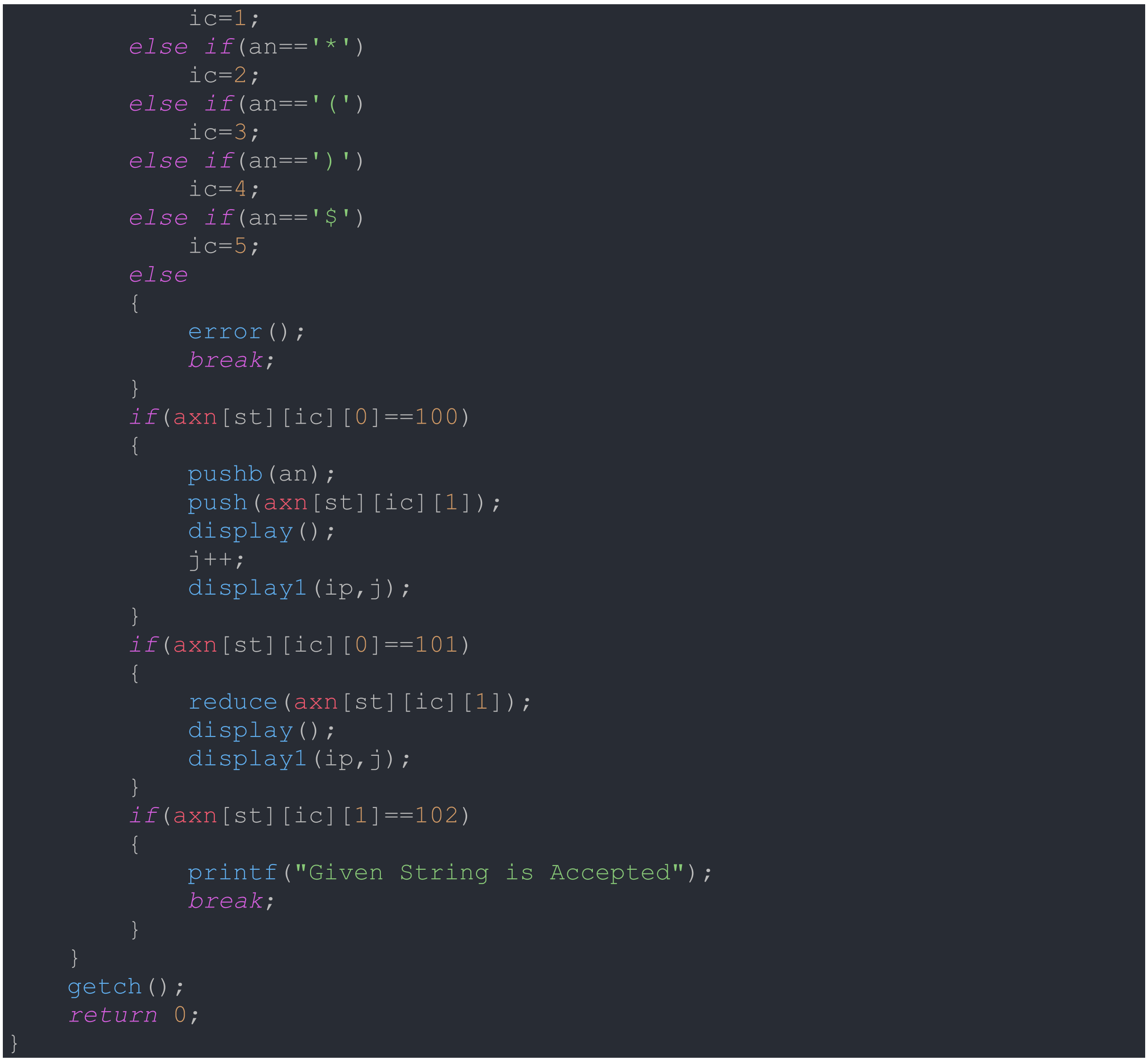
```
//To write a code for LR(0) Parser for following Production:
//E->E+T
//          T->T*F/F
//          F->(E)/char
#include<string.h>
#include<conio.h>
#include<stdio.h>
int axn[][6][2]={
        {{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},
```



```
        {{-1,-1},{100,6},{-1,-1},{-1,-1},{-1,-1},{102,102}},
        {{-1,-1},{101,2},{100,7},{-1,-1},{101,2},{101,2}},
        {{-1,-1},{101,4},{101,4},{-1,-1},{101,4},{101,4}},
        {{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},
        {{100,5},{101,6},{101,6},{-1,-1},{101,6},{101,6}},
        {{100,5},{-1,-1},{-1,-1},{-1,-1},{-1,-1},{-1,-1}},
        {{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},
        {{-1,-1},{100,6},{-1,-1},{-1,-1},{100,11},{-1,-1}},
        {{-1,-1},{101,1},{100,7},{-1,-1},{101,1},{101,1}},
        {{-1,-1},{101,3},{101,3},{-1,-1},{101,3},{101,3}},
        {{-1,-1},{101,5},{101,5},{-1,-1},{101,5},{101,5}}
};
int gotot[12][3]={1,2,3,-1,-1,-1,-1,-1,-1,-1,-1,-1,8,2,3,-1,-1,-1,-1,
                  9,3,-1,-1,10,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1};
int a[10];
char b[10];
int top=-1,btop=-1,i;
void push(int k)
{
    if(top<9)
        a[++top]=k;
}
void pushb(char k)
{
    if(btop<9)
        b[++btop]=k;
}
char TOS()
{
    return a[top];
}
void pop()
{
    if(top>=0)
        top--;
}
void popb()
{
    if(btop>=0)
        b[btop--]='\0';
}
void display()
{
    for(i=0;i<=top;i++)
        printf("%d%c",a[i],b[i]);
}
void display1(char p[],int m)
{
    int l;
    printf("\t\t");
    for(l=m;p[l]!='\0';l++)
        printf("%c",p[l]);
    printf("\n");
}
void error()
{
    printf("\n\nSyntax Error");
}
void reduce(int p)
{
    int len,k,ad;
```



```
    char src,*dest;
    switch(p)
    {
        case 1:dest="E+T";
            src='E';
            break;
        case 2:dest="T";
            src='E';
            break;
        case 3:dest="T*F";
            src='T';
            break;
        case 4:dest="F";
            src='T';
            break;
        case 5:dest="(E)";
            src='F';
            break;
        case 6:dest="i";
            src='F';
            break;
        default:dest="\0";
            src='\0';
            break;
    }
    for(k=0;k<strlen(dest);k++)
    {
        pop();
        popb();
    }
    pushb(src);
    switch(src)
    {
        case 'E': ad=0;
            break;
        case 'T': ad=1;
            break;
        case 'F': ad=2;
            break;
        default: ad=-1;
            break;
    }
    push(gotot[TOS()][ad]);
}
int main()
{
    int j,st,ic;
    char ip[20]="\0",an;
    clrscr();
    printf("Enter any String :- ");
    gets(ip);
    push(0);
    display();
    printf("\t%s\n",ip);
    for(j=0;ip[j]!='\0';)
    {
        st=TOS();
        an=ip[j];
        if(an>='a'&an<='z')
            ic=0;
        else if(an=='+')
```



```
            ic=1;
        else if(an=='*')
            ic=2;
        else if(an=='(')
            ic=3;
        else if(an==')')
            ic=4;
        else if(an=='$')
            ic=5;
        else
        {
            error();
            break;
        }
        if(axn[st][ic][0]==100)
        {
            pushb(an);
            push(axn[st][ic][1]);
            display();
            j++;
            display1(ip,j);
        }
        if(axn[st][ic][0]==101)
        {
            reduce(axn[st][ic][1]);
            display();
            display1(ip,j);
        }
        if(axn[st][ic][1]==102)
        {
            printf("Given String is Accepted");
            break;
        }
    }
    getch();
    return 0;
}
```

Output:

```
Enter any String :- a+b*c
0 a+b*c
0a5 +b*c
0F3 +b*c
0T2 +b*c
0E1 +b*c
0E1+6 b*c
0E1+6b5 *c
0E1+6F3 *c
0E1+6T9 *c
0E1+6T9*7 c
0E1+6T9*7c5
```