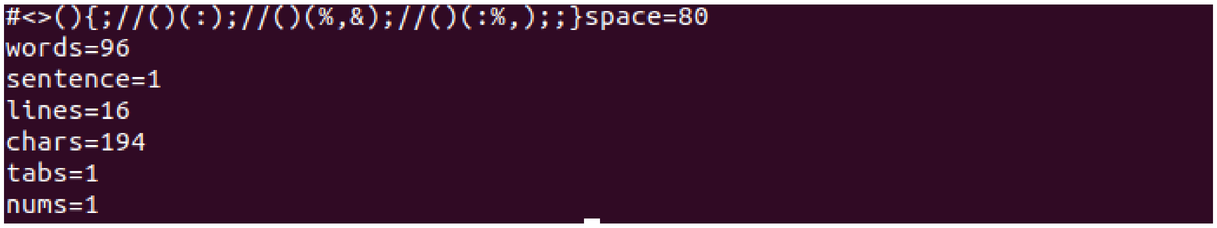
Exp 1

```
%{
int w=0,s=0, space=0,chars=0,n=0,tab=0,line=0,dig=0;
%}

%%
[" "] {space++;}
['.'] {s++;}
[\t] {tab++;}
[\n] {line++;}
[a-zA-Z] {chars++;}
[0-9]* {n++;}
%%

int main()
{
yyin=fopen("spcc1.c","r");
yylex();
int word = space+line+tab-n;
printf("space=%d\n",space-tab/2);
printf("words=%d\n",word);
printf("sentence=%d\n",s);
printf("lines=%d\n",line);
printf("chars=%d\n",chars);
printf("tabs=%d\n",tab);
printf("nums=%d\n",n);
return 0;
}
```

INPUT FILE

```
#include <stdio.h>
int main()
{
    int number;

    // printf() dislpays the formatted output
    printf("Enter an integer: ");

    // scanf() reads the formatted input and stores them
    scanf("%d", &number);

    // printf() displays the formatted output
    printf("You entered: %d", number);
    return 0;
}
//OUTPUT
```



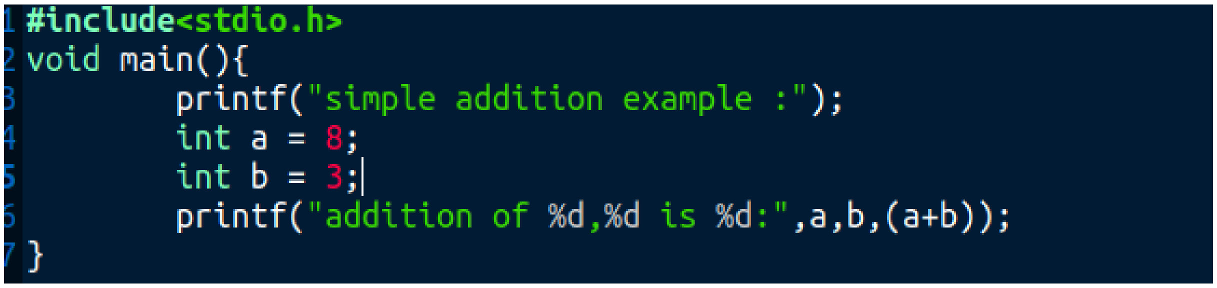
```
#<>(){;//()(:);//()(%,&);//()(:%,);;}space=80
words=96
sentence=1
lines=16
chars=194
tabs=1
nums=1
```

Exp 2

```
%{
#include<stdio.h>
int LOOKUP = 0;
int state;
int count=0;
int add_word(char *word) ;
int lookup_word(char *word);
%}
%%
\/\/.* {printf("%s Comment\n", yytext);}
\/\*([a-zA-z]*|(\n)*)*\*\/ {printf("%s Multiline comment\n", yytext);}
#include["<][a-zA-Z.]+[">] { printf("%s include statement\n", yytext); }
int|main|return|void|printf           { printf("%s is a keyword\n", yytext); }
\{|\(       { printf("%s opening brace\n", yytext);}
\}|\)       { printf("%s closing brace\n", yytext);}
\ |\,|\|\t|\;             {}
\n {state=LOOKUP;}
[a-zA-Z][a-zA-Z0-9_]* {
                                    add_word(yytext);
            }
\+|\-|\*|\\|=|\<|\> { printf("%s Operator\n", yytext ); }
\-?(([0-9]+)|([0-9]+\.[0-9]+)) {printf("%s Number\n", yytext);}
%%
int main(){
    yyin=fopen("program.c","r");
            yylex();
return 0;
}

int yywrap()
{
return 1;}

struct word{
            char *word_name;
            int count;
            struct word *next;
};
struct word *word_list;
int add_word(char *word){
            struct word *wp;
            int _count = lookup_word(word);
            if(_count!=LOOKUP){
                        printf("%s Identifier%d\n",word, _count);
                        return 0;
            }
            count++;
            wp = (struct word *)malloc(sizeof(struct word));
            wp->next = word_list;
            wp->word_name = (char *)malloc(strlen(word)+1);
            strcpy(wp->word_name,word);
            wp->count=count;
            word_list=wp;
            printf("%s added to word list as identifier%d\n", word, count);
            return 1;
}
```

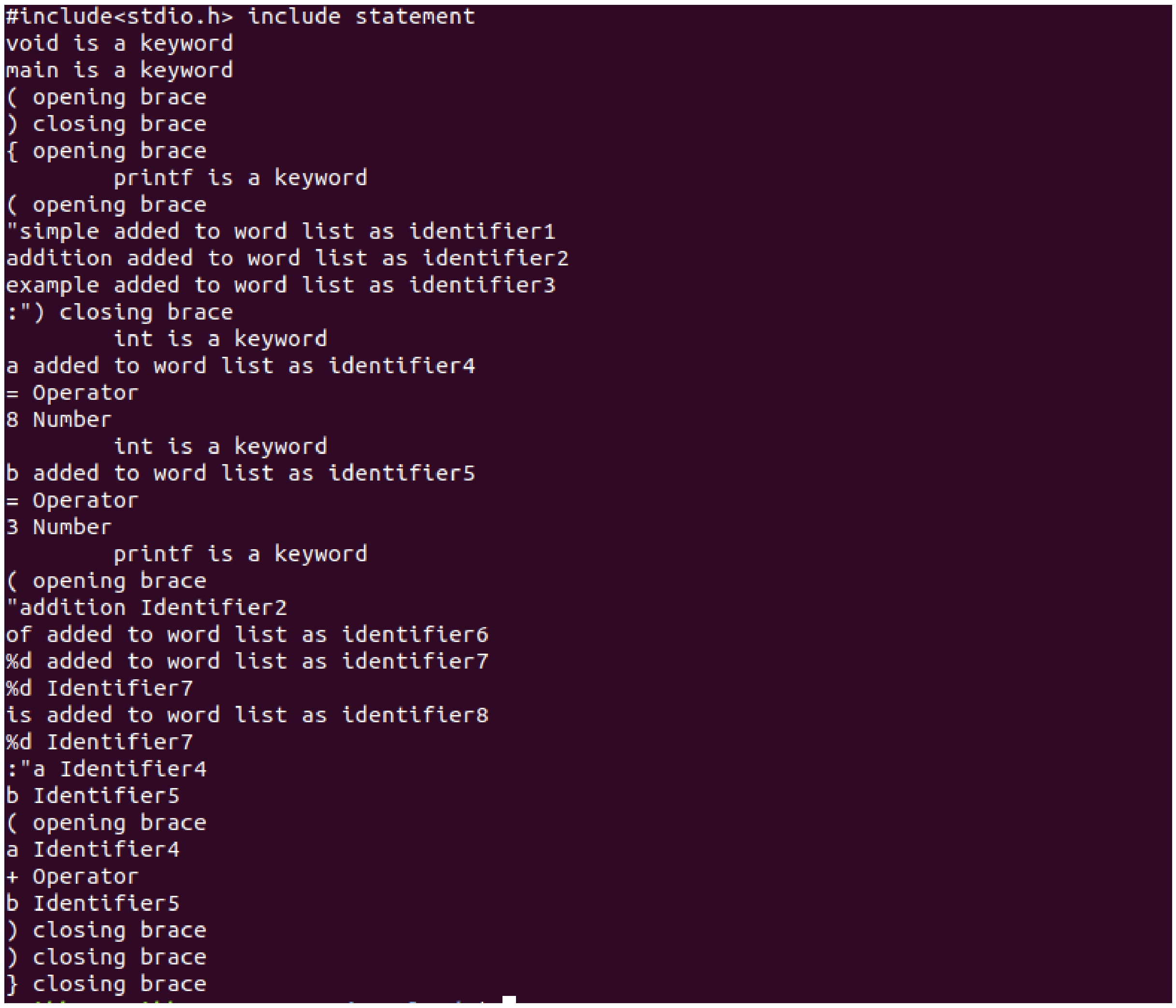
```
int lookup_word(char *word){
	struct word *wp=word_list;
	while(wp){
		if(strcmp(wp->word_name,word)==0){
			return wp->count;
		}
		wp = wp->next;
	};
	return LOOKUP;
}
```

// C PROGRAM FOR INPUT ANALYSIS



```c
#include<stdio.h>
void main(){
        printf("simple addition example :");
        int a = 8;
        int b = 3;
        printf("addition of %d,%d is %d:",a,b,(a+b));
}
```

//OUTPUT



```
#include<stdio.h> include statement
void is a keyword
main is a keyword
( opening brace
) closing brace
{ opening brace
        printf is a keyword
( opening brace
"simple added to word list as identifier1
addition added to word list as identifier2
example added to word list as identifier3
:") closing brace
        int is a keyword
a added to word list as identifier4
= Operator
8 Number
        int is a keyword
b added to word list as identifier5
= Operator
3 Number
        printf is a keyword
( opening brace
"addition Identifier2
of added to word list as identifier6
%d added to word list as identifier7
%d Identifier7
is added to word list as identifier8
%d Identifier7
:"a Identifier4
b Identifier5
( opening brace
a Identifier4
+ Operator
b Identifier5
) closing brace
) closing brace
} closing brace
```

Exp 3

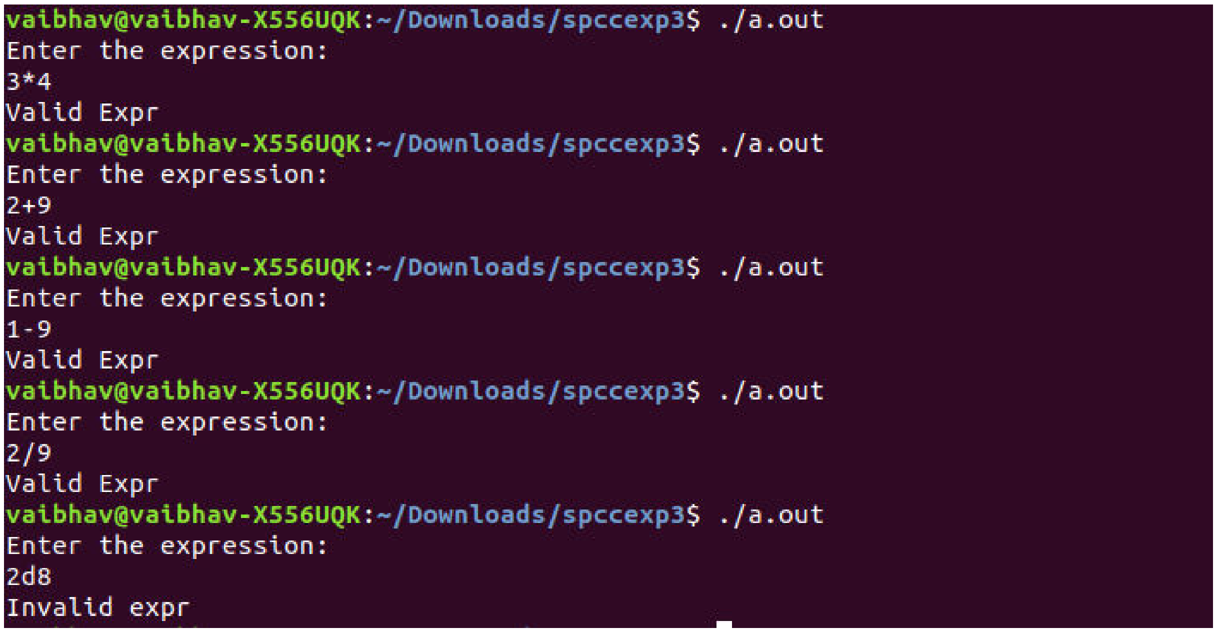
YACC FILE > “exp3.y”

```
%{
	#include<stdio.h>
%}

%token ID NUMBER

%left '+' '-'
%left '*' '/'

%%
stmt:expr
;
expr: expr'+'expr
|  expr'-'expr
|  expr'*'expr
|  expr'/'expr
| NUMBER
| ID
|
;
%%

void main()
{
printf("Enter the expression:\n");
yyparse();
printf("Valid Expr\n");
exit(0);
}
void yyerror()
{
printf("Invalid expr\n");
exit(0);
}
```

FLEX FILE > “exp3.l”

```
%{
 #include "y.tab.h"
%}

%%
[0-9] {return ID;}
[a-zA-Z] {return NUMBER;}
[ \t] {;}
\n {return 0;}
. {return yytext[0];}
%%
```

OUTPUT:



```
vaibhav@vaibhav-X556UQK:~/Downloads/spccexp3$ ./a.out
Enter the expression:
3*4
Valid Expr
vaibhav@vaibhav-X556UQK:~/Downloads/spccexp3$ ./a.out
Enter the expression:
2+9
Valid Expr
vaibhav@vaibhav-X556UQK:~/Downloads/spccexp3$ ./a.out
Enter the expression:
1-9
Valid Expr
vaibhav@vaibhav-X556UQK:~/Downloads/spccexp3$ ./a.out
Enter the expression:
2/9
Valid Expr
vaibhav@vaibhav-X556UQK:~/Downloads/spccexp3$ ./a.out
Enter the expression:
2d8
Invalid expr
```

Exp 4:

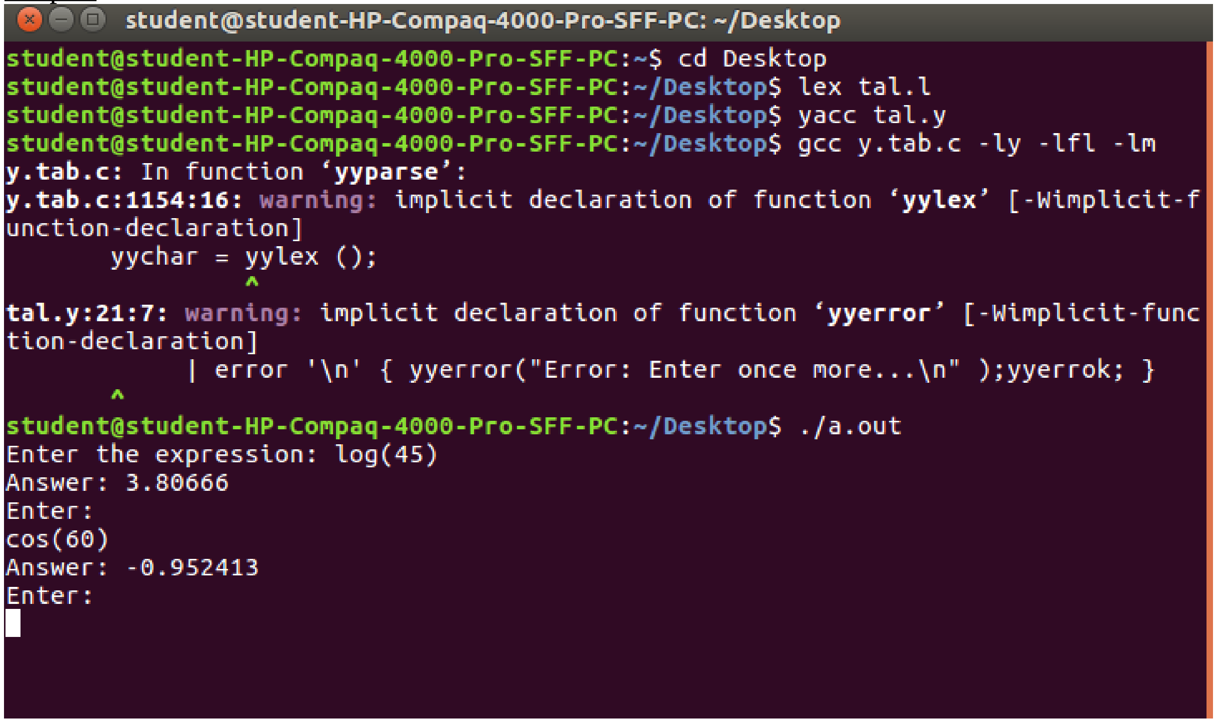
YACC Program:

```
%{
   #include<ctype.h>
   #include<stdio.h>
#include <math.h>

   #define YYSTYPE double
%}

%token NUM
%token COS SIN TAN LOG

%left '+' '-'
%left '*' '/'
%right UMINUS

%%

S        : S E '\n' { printf("Answer: %g \nEnter:\n", $2); }
         | S '\n'
         |
         | error '\n' { yyerror("Error: Enter once more...\n" );yyerrok; }
         ;
E        : E '+' E    { $$ = $1 + $3; }
         | E'-'E    { $$=$1-$3; }
         | E'*'E    { $$=$1*$3; }

          |E'/'E    { $$=$1/$3; }
         | '('E')'    { $$=$2; }
         | '-'E %prec UMINUS { $$= -$2; }
         | NUM
         | COS'('E')' {$$=cos($3);}
         | SIN'('E')' {$$=sin($3);}
         | TAN'('E')' {$$=tan($3);}
         | LOG'('E')' {$$=log($3);}

         ;

%%

#include "lex.yy.c"

int main()
{
   printf("Enter the expression: ");
   yyparse();
}
```

LEX Program:

```
%{

#include <math.h>
%}

DIGIT [0-9]+\.?|[0-9]*\.[0-9]+

%%

{DIGIT}    {yylval=atof(yytext);return NUM;}
cos|COS {return COS;}
sin|SIN {return SIN;}
tan|TAN {return TAN;}
log|LOG {return LOG;}
\n|.    {return yytext[0];}
```

Output:



```
student@student-HP-Compaq-4000-Pro-SFF-PC: ~/Desktop
student@student-HP-Compaq-4000-Pro-SFF-PC:~$ cd Desktop
student@student-HP-Compaq-4000-Pro-SFF-PC:~/Desktop$ lex tal.l
student@student-HP-Compaq-4000-Pro-SFF-PC:~/Desktop$ yacc tal.y
student@student-HP-Compaq-4000-Pro-SFF-PC:~/Desktop$ gcc y.tab.c -ly -lfl -lm
y.tab.c: In function ‘yyparse’:
y.tab.c:1154:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-f
unction-declaration]
       yychar = yylex ();
                ^
tal.y:21:7: warning: implicit declaration of function ‘yyerror’ [-Wimplicit-func
tion-declaration]
            | error '\n' { yyerror("Error: Enter once more...\n" );yyerrok; }
       ^
student@student-HP-Compaq-4000-Pro-SFF-PC:~/Desktop$ ./a.out
Enter the expression: log(45)
Answer: 3.80666
Enter:
cos(60)
Answer: -0.952413
Enter:
```

Exp 6 ICG

“teach5a.l” file:-

```
%{
	#include"y.tab.h"
%}
%%
[a-zA-Z]+	{strcpy(yylval.str,yytext);   return Var;}
[0-9]+		{strcpy(yylval.str,yytext);   return Num;}
\n		{return 0;}
.		{return yytext[0];}
%%
int yywrap()
{
	return 1;
}
```

“teach5a.y” file:-

```
%{
	#include<stdio.h>
	#include<stdlib.h>
	#include<string.h>
	char * createT();
	int tempcount=0;
	int top=-1;
%}
%union
{
char str[30];
}
%left '+'
%left  '-'
%left '*'
%left  '/'
%token <str> Var
%token <str> Num
```

```
%type  <str> s
%type  <str> exp
%%
s:          Var '=' exp    {printf("\n%s=%s\n",$1,$3);}
exp:       '(' exp ')'       {strcpy($$,$2);}
           | exp '+' exp    {strcpy($$,createT());printf("\n%s=%s+%s",$$,$1,$3);}
           |exp '-' exp     {strcpy($$,createT());printf("\n%s=%s-%s",$$,$1,$3);}
           | exp '*' exp    {strcpy($$,createT());printf("\n%s=%s*%s",$$,$1,$3);}
           | exp '/' exp    {strcpy($$,createT());printf("\n%s=%s/%s",$$,$1,$3);}
           | Num            {strcpy($$,$1);}
           | Var            {strcpy($$,$1);}


%%

char * createT()
{
           char snum[30],*ptr;
           sprintf(snum,"t%d",tempcount);
           ptr=snum;
           tempcount++;
           return ptr;
}
int main()
{
           yyparse();
           return 0;

}
int yyerror(char *err)
{
           printf("\nInvalid");
           exit(0);
}
```

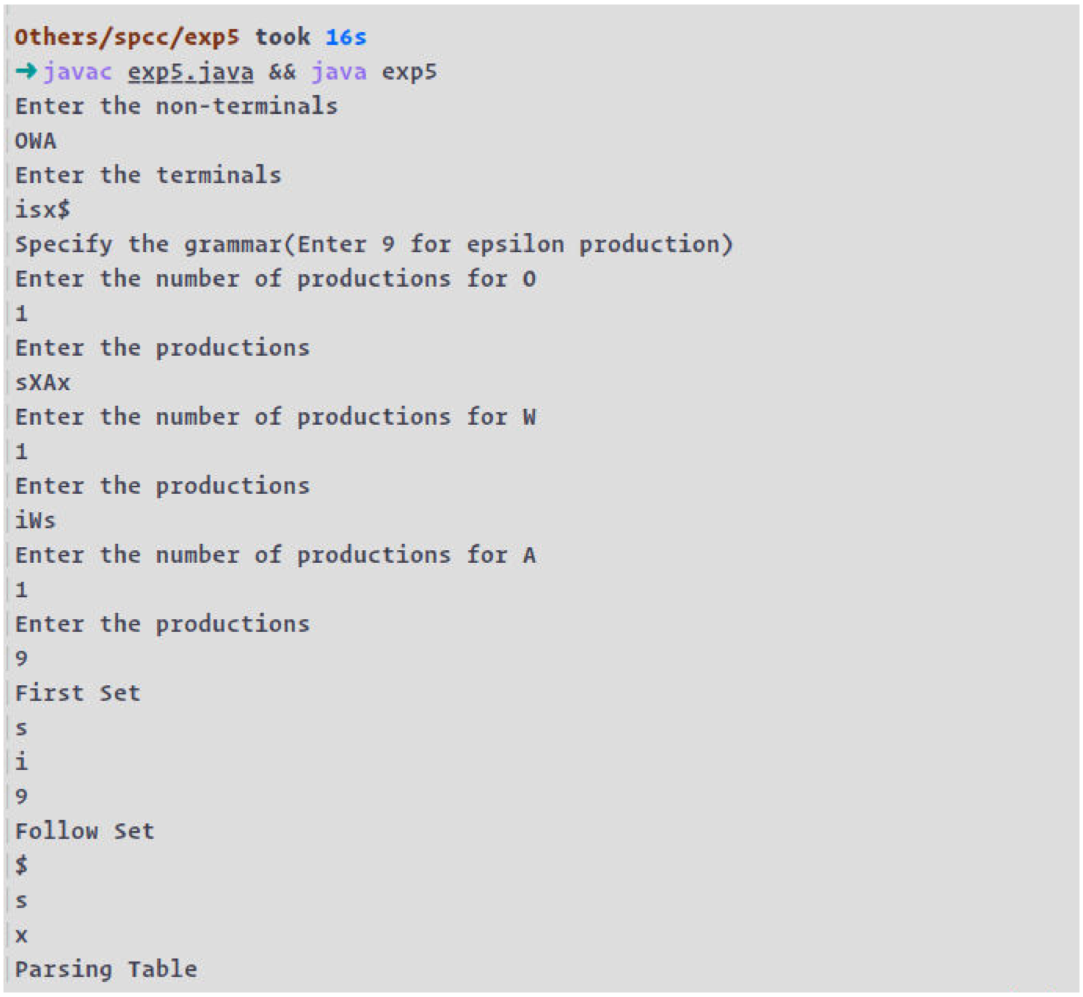
Output:

```
student@student-HP-Compaq-4000-Pro-SFF-PC:~/Desktop/exp5$ ./a.out
a=(b+c*d*e+f)

t0=c*d
t1=t0*e
t2=b+t1
t3=t2+f
a=t3


a=a+b

t0=a+b
a=t0



a=c+d*(e-f)

t0=e-f
t1=d*t0
t2=c+t1
a=t2
```

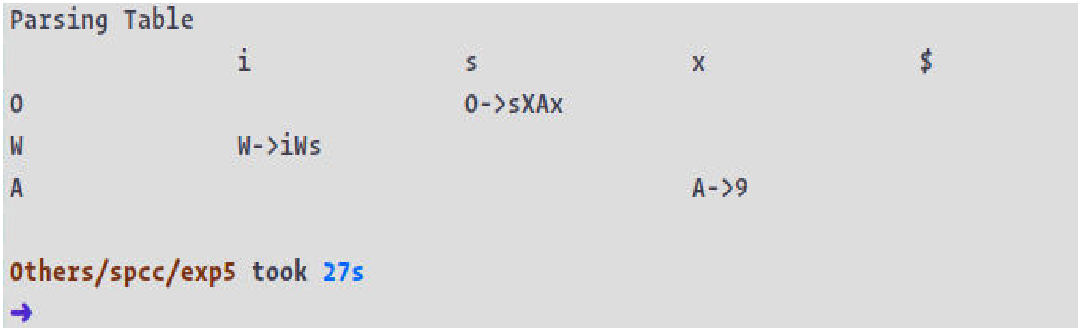
Exp5

```
import java.io.*;
import java.util.*;
class exp5 {
  static char ntermnl[],termnl[];
  static int ntlen,tlen;
  static String grmr[][],fst[],flw[];
  public static void main(String args[]) throws IOException {
    String nt,t;
    int i,j,n;
    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter the non-terminals");
    nt=br.readLine();
    ntlen=nt.length();
    ntermnl=new char[ntlen];
    ntermnl=nt.toCharArray();
    System.out.println("Enter the terminals");
    t=br.readLine();
    tlen=t.length();
    termnl=new char[tlen];
    termnl=t.toCharArray();
    System.out.println("Specify the grammar(Enter 9 for epsilon production)");
    grmr=new String[ntlen][];
    for(i=0;i<ntlen;i++) {
      System.out.println("Enter the number of productions for "+ntermnl[i]);
      n=Integer.parseInt(br.readLine());
      grmr[i]=new String[n];
      System.out.println("Enter the productions");
      for(j=0;j<n;j++)
        grmr[i][j]=br.readLine();
    }
    fst=new String[ntlen];
    for(i=0;i<ntlen;i++)
      fst[i]=first(i);
    System.out.println("First Set");
    for(i=0;i<ntlen;i++)
      System.out.println(removeDuplicates(fst[i]));
    flw=new String[ntlen];
    for(i=0;i<ntlen;i++)
      flw[i]=follow(i);
    System.out.println("Follow Set");
    for(i=0;i<ntlen;i++)
      System.out.println(removeDuplicates(flw[i]));

    System.out.println("Parsing Table");
    System.out.print("\t\t");
    for(i=0;i<tlen;i++){
      System.out.print(termnl[i] + "\t\t");
    }
  System.out.println("");
  for(i=0;i<ntlen;i++) {
    System.out.print(ntermnl[i] + "\t\t");
    String first = removeDuplicates(fst[i]);
    String follow = removeDuplicates(flw[i]);
    for(j=0; j<tlen; j++) {
      boolean hasEpsilon = first.contains("9");
      if(first.contains("" + termnl[j])) {
          System.out.print(ntermnl[i] + "->"+grmr[i][0]);
      }
      else System.out.print("");
      if(hasEpsilon) {
        if(follow.contains(""+ termnl[j])) {
          if(!first.contains("" +termnl[j]))
            System.out.print(ntermnl[i] + "->"+grmr[i][0] );
        }
      }
      System.out.print("\t\t");
    }
    System.out.println("");
  }
}
static String first(int i) {
  int j,k,l=0,found=0;
  String temp="",str="";
  for(j=0;j<grmr[i].length;j++)
  {for(k=0;k<grmr[i][j].length();k++,found=0)
    {for(l=0;l<ntlen;l++)
      {
        if(grmr[i][j].charAt(k)==ntermnl[l]) {
          str=first(l);if(!(str.length()==1 && str.charAt(0)=='9'))
            temp=temp+str;
          found=1;
          break;}}
      if(found==1)
      {
        if(str.contains("9"))
          continue;
      }
      else
        temp=temp+grmr[i][j].charAt(k);
      break;}}
  return temp;
}
static String follow(int i)
{
  char pro[],chr[];
  String temp="";
  int j,k,l,m,n,found=0;
  if(i==0)
    temp="$";
  for(j=0;j<ntlen;j++)
  {for(k=0;k<grmr[j].length;k++)
    {
      pro=new char[grmr[j][k].length()];
      pro=grmr[j][k].toCharArray();
      for(l=0;l<pro.length;l++)
      {if(pro[l]==ntermnl[i])
        {
          if(l==pro.length-1)
          {
            if(j<i)
              temp=temp+flw[j];
          }
          else
          {
            for(m=0;m<ntlen;m++)
            {
              if(pro[l+1]==ntermnl[m])
              {
                chr=new char[fst[m].length()];
                chr=fst[m].toCharArray();
                for(n=0;n<chr.length;n++)
                {
                  if(chr[n]=='9')
                  {
                    if(l+1==pro.length-1)
                      temp=temp+follow(j);
                    else
                      temp=temp+follow(m);
                  }
                  else
                    temp=temp+chr[n];
                }
                found=1;
              }}
            if(found!=1)
              temp=temp+pro[l+1];
          }}}}}return temp;
}
static String removeDuplicates(String str)
{
  int i;
  char ch;
  boolean seen[] = new boolean[256];
  StringBuilder sb = new StringBuilder(seen.length);
  for(i=0;i<str.length();i++)
  {
    ch=str.charAt(i);
    if (!seen[ch])
    {
      seen[ch] = true;
      sb.append(ch);
    }}
  return sb.toString();
}}
```

OUTPUT:



```
Others/spcc/exp5 took 16s
➜ javac exp5.java && java exp5
Enter the non-terminals
OWA
Enter the terminals
isx$
Specify the grammar(Enter 9 for epsilon production)
Enter the number of productions for O
1
Enter the productions
sXAx
Enter the number of productions for W
1
Enter the productions
iWs
Enter the number of productions for A
1
Enter the productions
9
First Set
s
i
9
Follow Set
$
s
x
Parsing Table
```

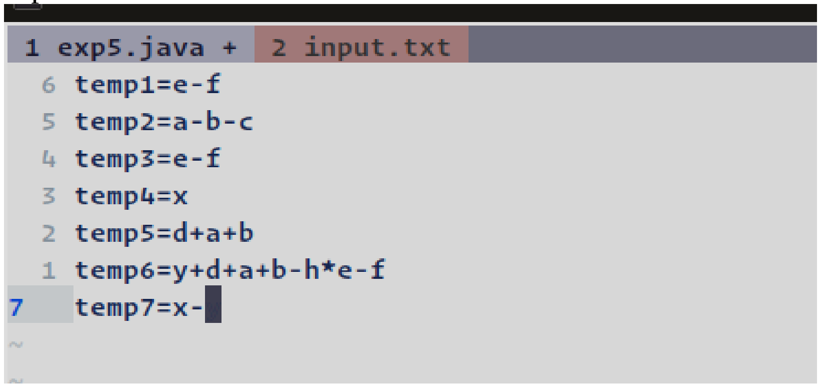


```
Parsing Table
                i               s               x               $
O                               O->sXAx
W               W->iWs
A                                               A->9

Others/spcc/exp5 took 27s
➜
```

Exp7

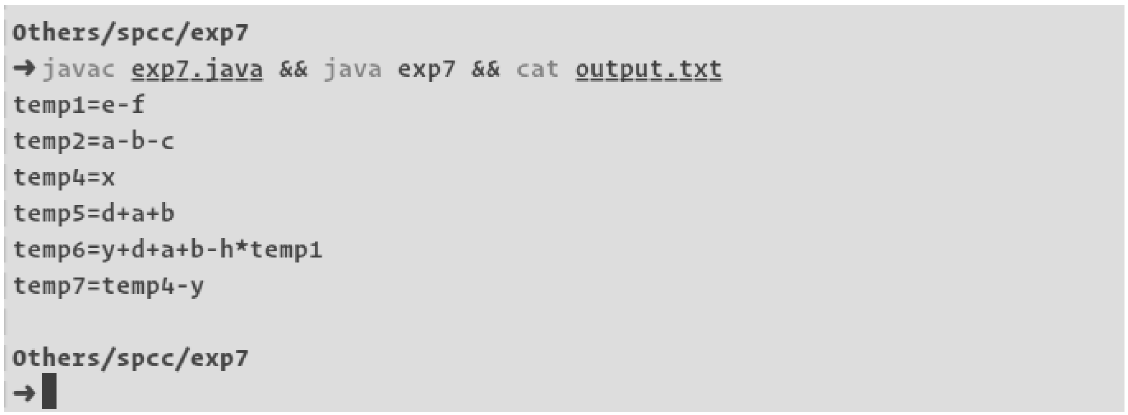
```
import java.io.*;
import java.util.*;
class exp7{
  public static void main(String args[])throws IOException  {
    String s,temp;
    String arr[][]=new String[10][2];
    int flag=0,index=0;
    BufferedReader br=new BufferedReader(new InputStreamReader(new
          FileInputStream("input.txt")));
    File op = new File("output.txt");
    if (!op.exists())
      op.createNewFile();
    BufferedWriter output = new BufferedWriter(new FileWriter(op.getAbsoluteFile()));
    for(;(s=br.readLine())!=null;flag=0)    {
      temp=s.substring(s.indexOf("=")+1);
      for(int i=0;i<index;i++)      {
        if(temp.equals(arr[i][1]))        {
          flag=1;
          break;
        }
        else if(temp.contains(arr[i][1]))
          s=s.replaceAll(arr[i][1],arr[i][0]);
      }
      if(flag==0)      {
        arr[index][0]=s.substring(0,s.indexOf("="));
        arr[index][1]=temp;
        index++;
        output.write(s);
        output.newLine();
      }
    }
    output.close();
  }
}
```

Input:



```
1 exp5.java +  2 input.txt
temp1=e-f
temp2=a-b-c
temp3=e-f
temp4=x
temp5=d+a+b
temp6=y+d+a+b-h*e-f
temp7=x-
```

Output

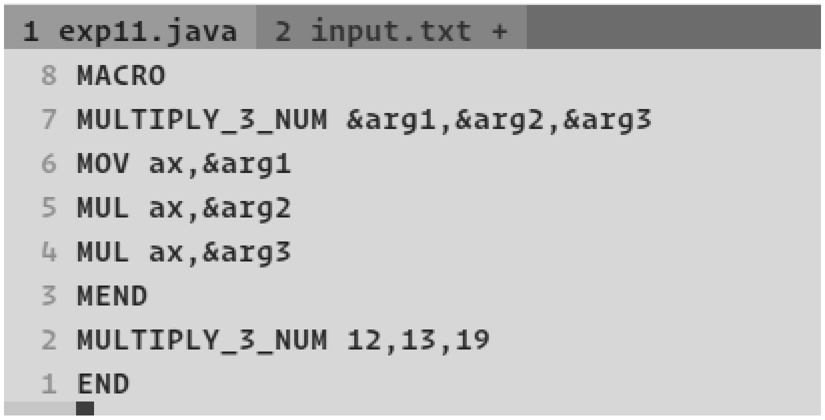


```
Others/spcc/exp7
➜ javac exp7.java && java exp7 && cat output.txt
temp1=e-f
temp2=a-b-c
temp4=x
temp5=d+a+b
temp6=y+d+a+b-h*temp1
temp7=temp4-y

Others/spcc/exp7
➜
```

Exp11

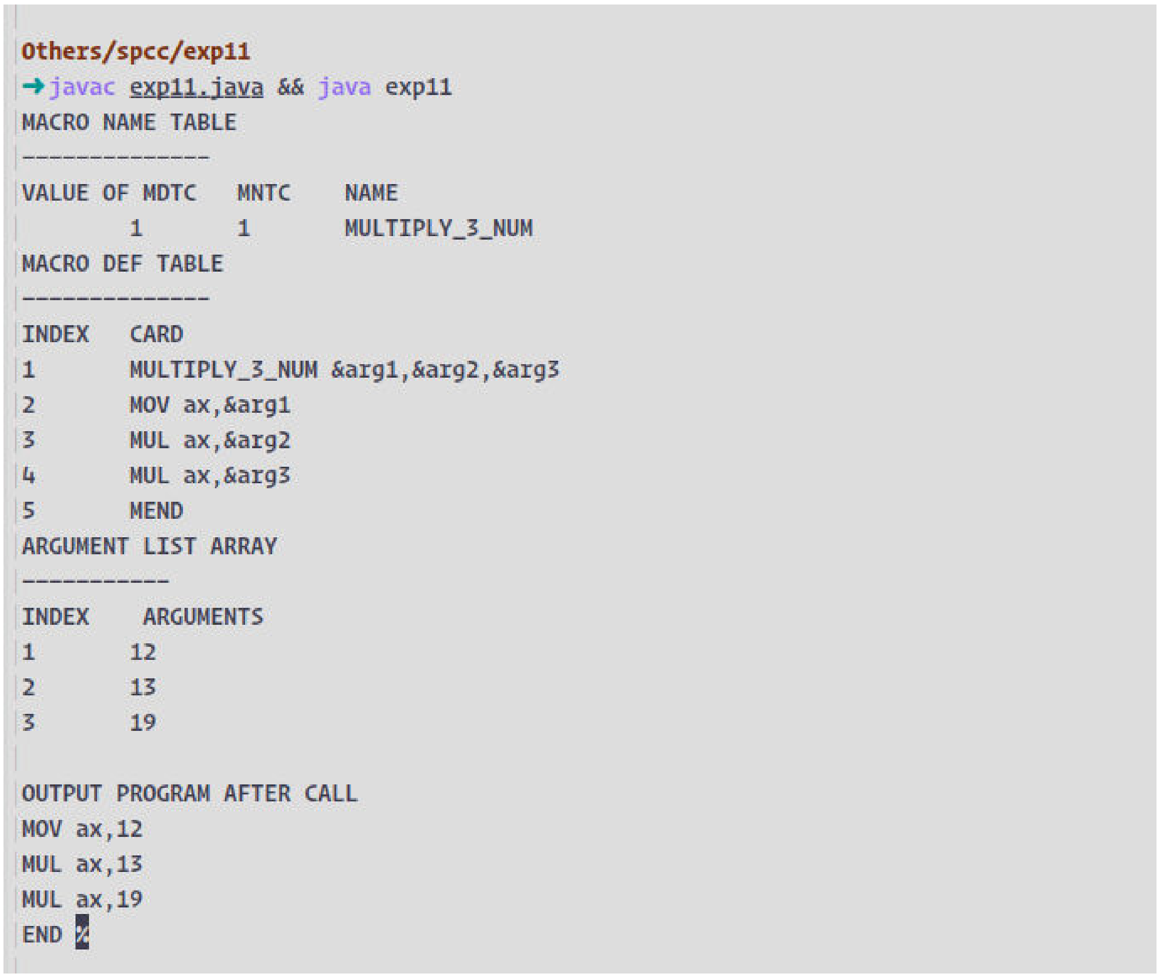
```
import java.io.*;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.Arrays;

public class exp11 {
  public static void main(String args[])throws IOException{
    int MDTC=1;
    int MNTC=1;
    int index=1;
    int macroindex=0;
    String arg[]=new String[10];
    String mname[]=new String[10];
    String MNT [][]=new String[10][10];
    String MDT [][]=new String[10][10];
    String output =new Scanner(new File("input.txt")).useDelimiter("\\Z").next();
    String result[]=output.split("\n");
    String result1[]=output.split("[,\\s\\?]");
    for(int k=0;k<result1.length;k++)    {
      if(result1[k].equals("MACRO")||result1[k].equals("macro"))      {
        mname[macroindex]=result1[k+1];
        macroindex++;
      }
    }
    System.out.println("\nMACRO NAME TABLE\n——————————————————————————————");
    System.out.println("VALUE OF MDTC\tMNTC\tNAME");
    for(int k=0;k<macroindex;k++)    {
      System.out.println("\t"+MDTC+"\t"+MNTC+"\t"+mname[k]);
      MNTC=MNTC+1;
    }
    System.out.println("\n\nMACRO DEF TABLE\n——————————————————————————————");
    System.out.println("INDEX\tCARD");
    for(int i=1;i<result.length;i++)    {
      System.out.println(MDTC+"\t"+result[i]);
      MDTC=MDTC+1;
      if(result[i].equals("MEND"))
        break;
    }
    System.out.print("\n\nARGUMENT LIST ARRAY\n———————————————————————");
    for(int k=3;k<result1.length;k++)    {
      if(result1[k].equals(mname[0]))      {
        arg[0]=result1[k+1];
        arg[1]=result1[k+2];
        arg[2]=result1[k+3];
      }    }
    System.out.println("\nINDEX\t ARGUMENTS");
    System.out.println("\n"+index+"\t"+arg[0]+"\n"+(index+1)+"\t"+arg[1]+"\n"+
(index+2)+"\t"+arg[2]+"\n");
    System.out.print("\n\nOUTPUT PROGRAM AFTER CALL\n");
    boolean inMacro = false;
    for(int i=0; i<result.length; i++) {
      String[] tokens = result[i].split("[,\\s\\?]");
      boolean macroCall = false;
      int argCounter = 0;
      for(String token: tokens) {
        if(token.equals("MACRO")){
          inMacro = true;
        }
        else if(token.equals("MEND")){
          inMacro = false;
        }
        else {
          if(!inMacro && Arrays.asList(mname).contains(token)) {
            macroCall = true;
            argCounter = 0;
          }
          else if(!macroCall && !inMacro) {
            System.out.print(token + " ");
          }
          else if(macroCall) {
            arg[argCounter++] = token;
          }
        }
      }
      if(macroCall) {
        macroCall = false;
        for(int j=2;i<result.length; j++) {
          if(result[j].equals("MEND"))
            break;
          System.out.println(result[j].replaceAll("&arg1", arg[0]).replaceAll("&arg2",
arg[1]).replaceAll("&arg3", arg[2]));
        }
      }
    }
  }
}
```

Input:



```
1 exp11.java   2 input.txt +
MACRO
MULTIPLY_3_NUM &arg1,&arg2,&arg3
MOV ax,&arg1
MUL ax,&arg2
MUL ax,&arg3
MEND
MULTIPLY_3_NUM 12,13,19
END
```

Output:



```
Others/spcc/exp11
➜ javac exp11.java && java exp11
MACRO NAME TABLE
--------------
VALUE OF MDTC   MNTC    NAME
        1       1       MULTIPLY_3_NUM
MACRO DEF TABLE
--------------
INDEX   CARD
1       MULTIPLY_3_NUM &arg1,&arg2,&arg3
2       MOV ax,&arg1
3       MUL ax,&arg2
4       MUL ax,&arg3
5       MEND
ARGUMENT LIST ARRAY
-----------
INDEX    ARGUMENTS
1       12
2       13
3       19

OUTPUT PROGRAM AFTER CALL
MOV ax,12
MUL ax,13
MUL ax,19
END
```

Exp 9 & 10

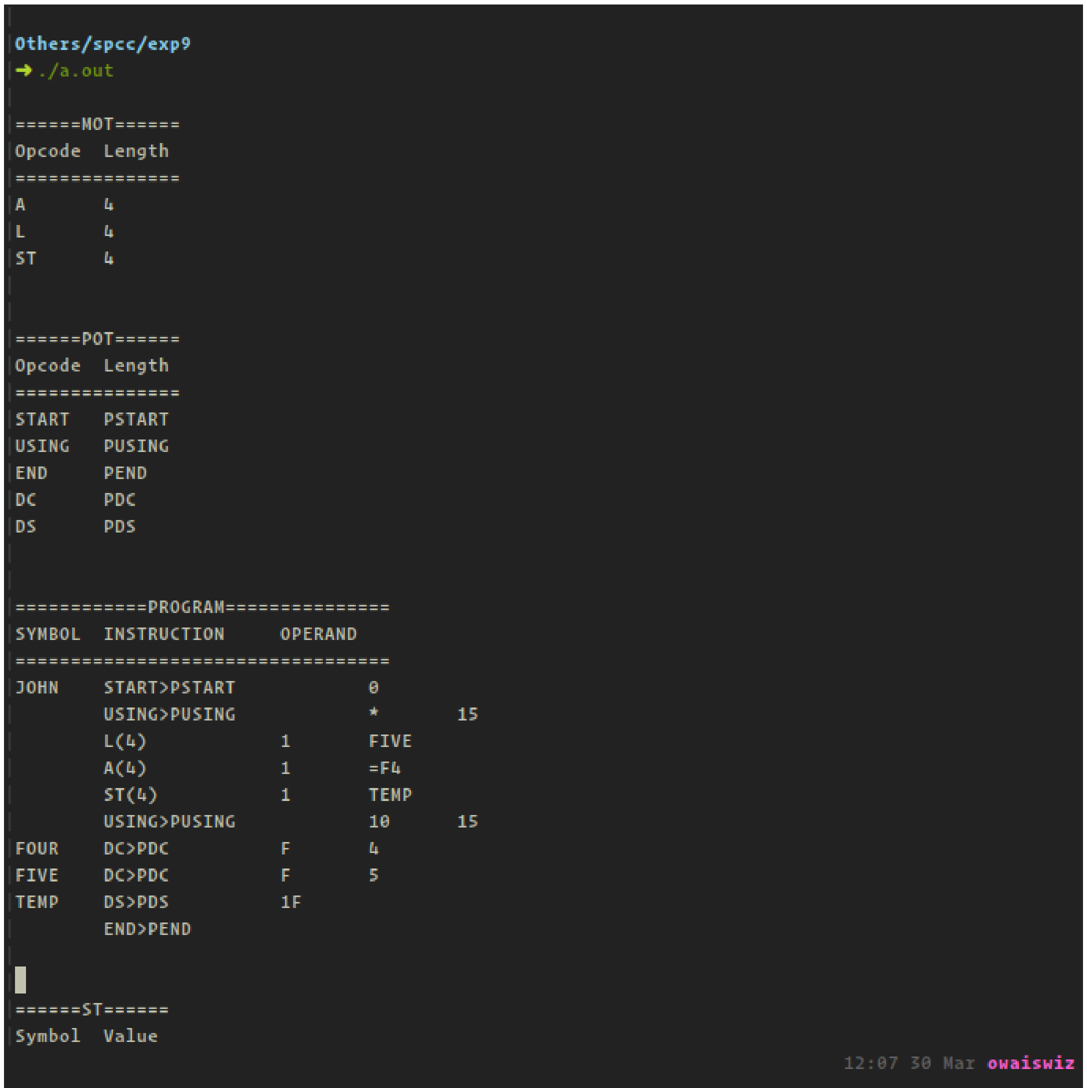
```
#include<stdio.h>
#include<string.h>
struct stomot{
  char opcode[10];
  int length;
};
struct stopot{
  char opcode[10];
  char routine[10];
};
struct stoprogram{
  char symbol[10];
  char instruction[10];
  char op1[10];
  char op2[10];
  int lc;
};
struct stopass{
  char symbol[10];
  char instruction[10];
  char op1[10];
  char op2[10];
  int lc;
};
struct stosymbol{
  char symbol[10];
  int value;
};
struct stoliteral{
  char symbol[10];
  int value;
};
struct stobase {
  char reg[10];
  char val[10];
};
void main(){
  int i,n,temp1,j=0,add=0,lc,k,flag=0,motflag=0,temp=0,nst,l,m,foundinst,foundinlit;
  char ois[] = "(0,15)";
  char str[10];
  struct stomot mot[10];
  struct stopot pot[10];
  struct stoprogram program[30];
  struct stopass pass[30];
  struct stosymbol symbol[10];
  struct stoliteral literal[10];
  struct stobase base[10];
```

```
strcpy(mot[0].opcode,"A");
strcpy(mot[1].opcode,"L");
strcpy(mot[2].opcode,"ST");
mot[0].length=4;
mot[1].length=4;
mot[2].length=4;
printf("\n======MOT======\n");
printf("Opcode\tLength\n");
printf("===============\n");
for(i=0;i<3;i++) {
  printf("%s\t%d\n",mot[i].opcode,mot[i].length);
}

strcpy(pot[0].opcode,"START");
strcpy(pot[1].opcode,"USING");
strcpy(pot[2].opcode,"END");
strcpy(pot[3].opcode,"DC");
strcpy(pot[4].opcode,"DS");
strcpy(pot[0].routine,"PSTART");
strcpy(pot[1].routine,"PUSING");
strcpy(pot[2].routine,"PEND");
strcpy(pot[3].routine,"PDC");
strcpy(pot[4].routine,"PDS");
//strcpy()
printf("\n\n======POT======\n");
printf("Opcode\tLength\n");
printf("===============\n");
for(i=0;i<5;i++) {
  printf("%s\t%s\n",pot[i].opcode,pot[i].routine);
}

strcpy(program[0].symbol,"JOHN");
strcpy(program[0].instruction,"START");
strcpy(program[0].op1,"0");
strcpy(program[0].op2," ");
strcpy(program[1].symbol," ");
strcpy(program[1].instruction,"USING");
strcpy(program[1].op1,"*");
strcpy(program[1].op2,"15");
strcpy(program[2].symbol," ");
strcpy(program[2].instruction,"L");
strcpy(program[2].op1,"1");
strcpy(program[2].op2,"FIVE");
strcpy(program[3].symbol," ");
strcpy(program[3].instruction,"A");
strcpy(program[3].op1,"1");
strcpy(program[3].op2,"=F4");
strcpy(program[4].symbol," ");
```

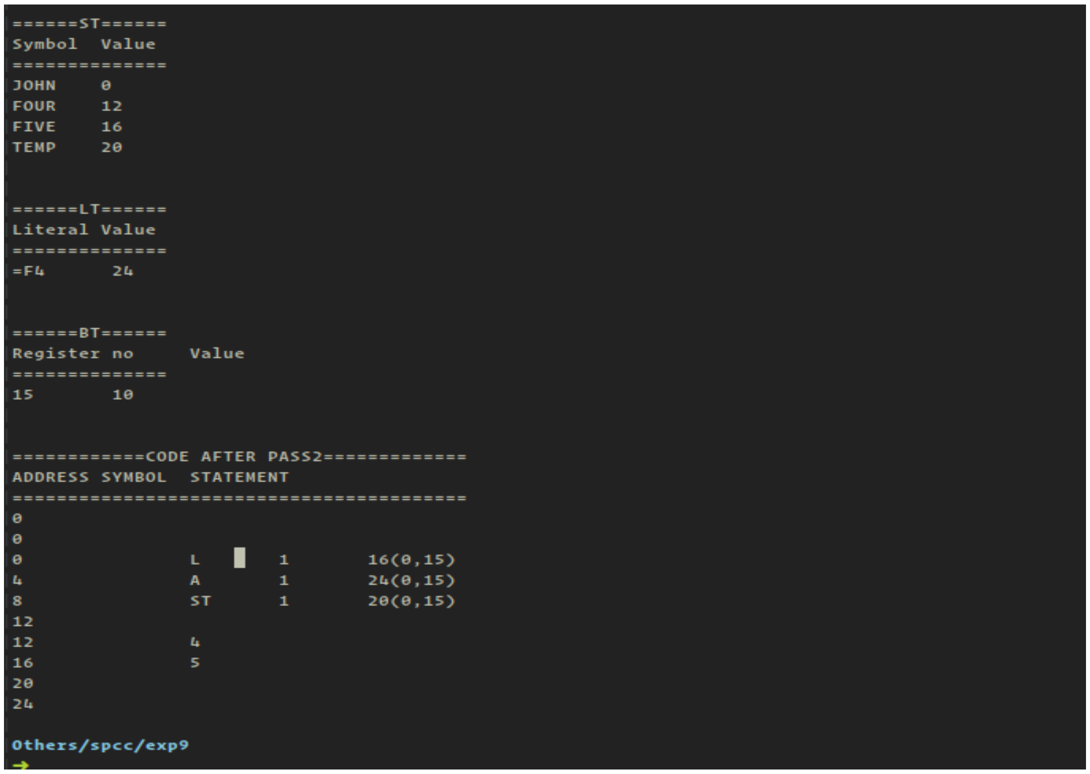
```
  strcpy(program[4].instruction,"ST");
  strcpy(program[4].op1,"1");
  strcpy(program[4].op2,"TEMP");
  strcpy(program[5].symbol," ");
  strcpy(program[5].instruction,"USING");
  strcpy(program[5].op1,"10");
  strcpy(program[5].op2,"15");
  strcpy(program[6].symbol,"FOUR");
  strcpy(program[6].instruction,"DC");
  strcpy(program[6].op1,"F");
  strcpy(program[6].op2,"4");
  strcpy(program[7].symbol,"FIVE");
  strcpy(program[7].instruction,"DC");
  strcpy(program[7].op1,"F");
  strcpy(program[7].op2,"5");
  strcpy(program[8].symbol,"TEMP");
  strcpy(program[8].instruction,"DS");
  strcpy(program[8].op1,"1F");
  strcpy(program[8].op2," ");
  strcpy(program[9].symbol," ");
  strcpy(program[9].instruction,"END");
  strcpy(program[9].op1," ");
  strcpy(program[9].op2," ");
  printf("\n\n============PROGRAM===============\n");
  printf("SYMBOL\tINSTRUCTION\tOPERAND\t\n");
  printf("==================================\n");
  for(i=0;i<10;i++) {
    temp1=0;
    for(j=0;j<3;j++){
      if((strcmp(mot[j].opcode,program[i].instruction))==0){
        temp=j;
        temp1=1;
        break;
      }
    }
    for(j=0;j<5;j++){
      if((strcmp(pot[j].opcode,program[i].instruction))==0){
        temp=j;
        temp1=0;
        break;
      }
    }
    if(temp1>0){
      printf("%s\t%s(%d)\t\t%s\t
%s\t\n",program[i].symbol,program[i].instruction,mot[temp].length,program[i].op1,program[i].op2);
    }else{
      printf("%s\t%s>%s\t\t%s\t
%s\t\n",program[i].symbol,program[i].instruction,pot[temp].routine,program[i].op1,program[i].op2);
    }
}
lc=0;
//symbol table
for(i=0;i<10;i++) {
  program[i].lc=lc;
  if(strcmp(program[i].symbol," ")!=0){
    strcpy(symbol[add].symbol,program[i].symbol);
    symbol[add].value=lc;
    add++;
  }
  for(j=0;j<3;j++) {
    if(strcmp(mot[j].opcode, program[i].instruction)==0) {
      lc=lc+mot[j].length;
      break;
    }
  }
  if(strcmp(program[i].instruction, "DC")==0 || strcmp(program[i].instruction, "DS")==0) {
    lc=lc+4;
  }
}
printf("\n\n======ST======\n");
printf("Symbol\tValue\n");
printf("==============\n");
for(i=0;i<add;i++) {
  printf("%s\t%d\n",symbol[i].symbol,symbol[i].value);
}
nst=add;
add=0;
for(i=0;i<10;i++) {
  if(program[i].op1[0]=='=') {
    strcpy(literal[add].symbol,program[i].op1);
    literal[add].value=lc;
    lc=lc+4;
    add++;
  }
  if(program[i].op2[0]=='=') {
    strcpy(literal[add].symbol,program[i].op2);
    literal[add].value=lc;
    lc=lc+4;
    add++;
  }
}
printf("\n\n======LT======\n");
printf("Literal\tValue\n");
printf("==============\n");
for(i=0;i<add;i++) {
  printf("%s\t %d\n",literal[i].symbol,literal[i].value);
}
```

```
for(i=0;i<10;i++) {
  if(strcmp(program[i].instruction,"USING")==0){
    if(strcmp(program[i].op1,"*")==0) {
      strcpy(base[0].val,"0");
    } else {
      strcpy(base[0].val,program[i].op1);
    }
    strcpy(base[0].reg,program[i].op2);
  }
}
printf("\n\n======BT======\n");
printf("Register no\tValue\n");
printf("=============\n");
printf("%s\t %s\n",base[0].reg,base[0].val);

for(i=0;i<10;i++) {
  if(strcmp(program[i].instruction,"DC")==0) {
    strcpy(pass[i].symbol," ");
    strcpy(pass[i].instruction,program[i].op2);
    strcpy(pass[i].op1," ");
    strcpy(pass[i].op2," ");
    pass[i].lc=program[i].lc;
  } else {
    motflag=0;
    for(j=0;j<3;j++) {
      if(strcmp(mot[j].opcode, program[i].instruction)==0) {
        motflag=1;
        strcpy(pass[i].symbol,program[i].symbol);
        strcpy(pass[i].instruction,program[i].instruction);
        flag=0;
        for(k=0;k<strlen(program[i].op1);k++) {
          if(!isdigit(program[i].op1[k])) {
            flag=1;
            break;
          }
        }
        if(flag==1) {
          foundinst=0;
          for(l=0;l<nst;l++) {
            if(strcmp(program[i].op1,symbol[l].symbol)==0) {
              foundinst=1;
              break;
            }
          }
          foundinlit=0;
          for(m=0;m<nst && foundinst==0;m++) {
            if(strcmp(program[i].op1,literal[m].symbol)==0) {
              foundinlit=1;
              break;
            }
          }
          if(foundinst==1) {
            sprintf(str, "%d", symbol[l].value);
            strcat(str,ois);
            strcpy(pass[i].op1,str);
          } else if(foundinlit==1) {
            sprintf(str, "%d", literal[m].value);
            strcat(str,ois);
            strcpy(pass[i].op1,str);
          } else {
            strcpy(pass[i].op2,"NotFound");
          }
        } else {
          strcpy(pass[i].op1,program[i].op1);
        }
        flag=0;
        for(k=0;k<strlen(program[i].op2);k++) {
          if(!isdigit(program[i].op2[k])) {
            flag=1;
            break;
          }
        }
        if(flag==1) {
          foundinst=0;
          for(l=0;l<nst;l++) {
            if(strcmp(program[i].op2,symbol[l].symbol)==0) {
              foundinst=1;
              break;
            }
          }
          foundinlit=0;
          for(m=0;m<nst && foundinst==0;m++) {
            if(strcmp(program[i].op2,literal[m].symbol)==0) {
              foundinlit=1;
              break;
            }
          }
          if(foundinst==1) {
            sprintf(str, "%d", symbol[l].value);
            strcat(str,ois);
            strcpy(pass[i].op2,str);
          } else if(foundinlit==1) {
            sprintf(str, "%d", literal[m].value);
            strcat(str,ois);
            strcpy(pass[i].op2,str);
          } else {
            strcpy(pass[i].op2,"NotFound");
          }
```

```
            } else {
              strcpy(pass[i].op2,program[i].op2);
            }
            pass[i].lc=program[i].lc;
            break;
          }
        }
        if(motflag==0) {
          strcpy(pass[i].symbol," ");
          strcpy(pass[i].instruction," ");
          strcpy(pass[i].op1," ");
          strcpy(pass[i].op2," ");
          pass[i].lc=program[i].lc;
        }
      }
    }
    printf("\n\n===========CODE AFTER PASS2============\n");
    printf("ADDRESS\tSYMBOL\tSTATEMENT\t\n");
    printf("=========================================\n");
    for(i=0;i<10;i++) {
      printf("%d\t%s\t%s\t%s\t%s\t\n",pass[i].lc,pass[i].symbol,pass[i].instruction,pass[i].op1,pass[i].op2);
    }
  }
```



```
Others/spcc/exp9
➜ ./a.out

======MOT======
Opcode  Length
===============
A       4
L       4
ST      4


======POT======
Opcode  Length
===============
START   PSTART
USING   PUSING
END     PEND
DC      PDC
DS      PDS


============PROGRAM===============
SYMBOL  INSTRUCTION     OPERAND
==================================
JOHN    START>PSTART            0
        USING>PUSING            *       15
        L(4)            1       FIVE
        A(4)            1       =F4
        ST(4)           1       TEMP
        USING>PUSING            10      15
FOUR    DC>PDC          F       4
FIVE    DC>PDC          F       5
TEMP    DS>PDS          1F
        END>PEND


======ST======
Symbol  Value
                                                        12:07 30 Mar owaiswiz
```



```
======ST======
Symbol  Value
==============
JOHN     0
FOUR     12
FIVE     16
TEMP     20


======LT======
Literal Value
==============
=F4      24


======BT======
Register no      Value
==============
15       10


============CODE AFTER PASS2=============
ADDRESS SYMBOL  STATEMENT
=========================================
0
0
0               L       1       16(0,15)
4               A       1       24(0,15)
8               ST      1       20(0,15)
12
12              4
16              5
20
24

Others/spcc/exp9
→
```