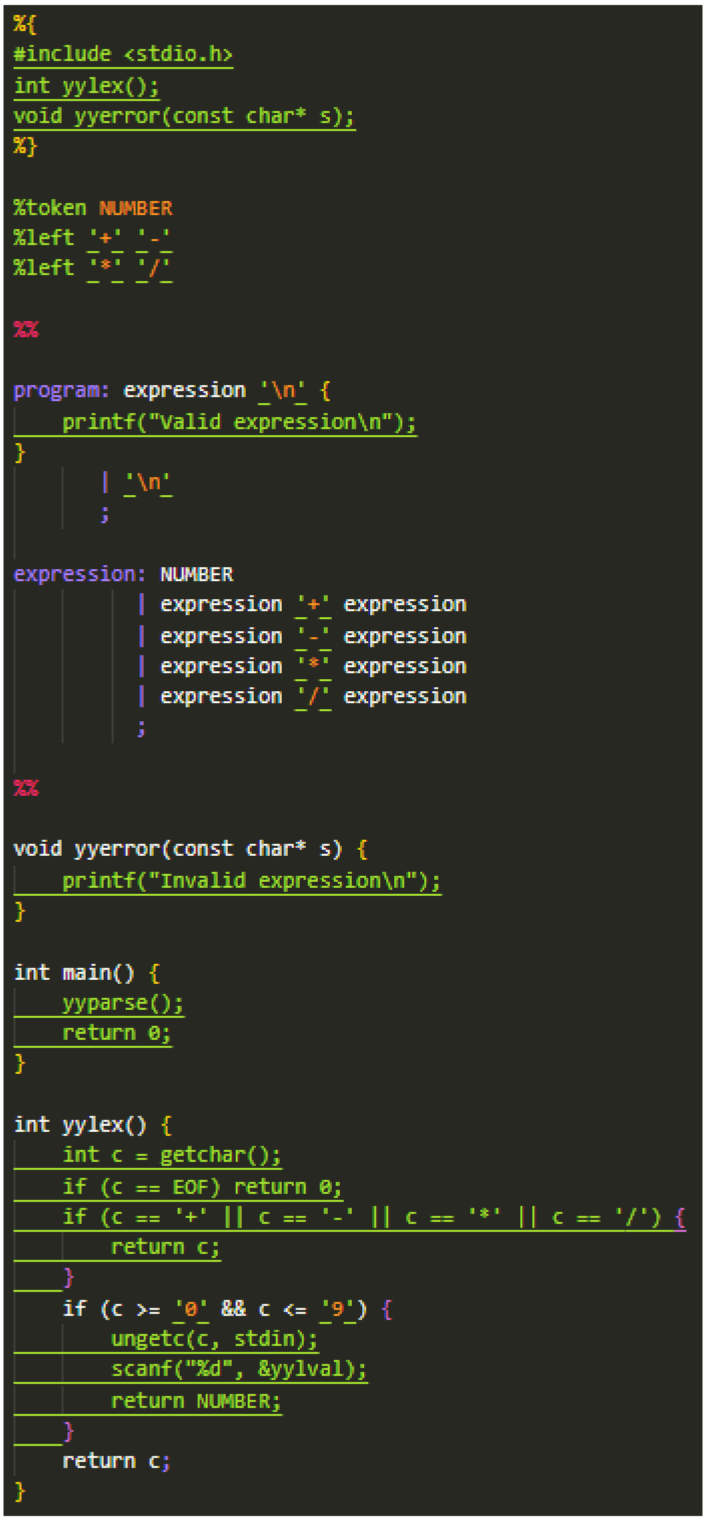
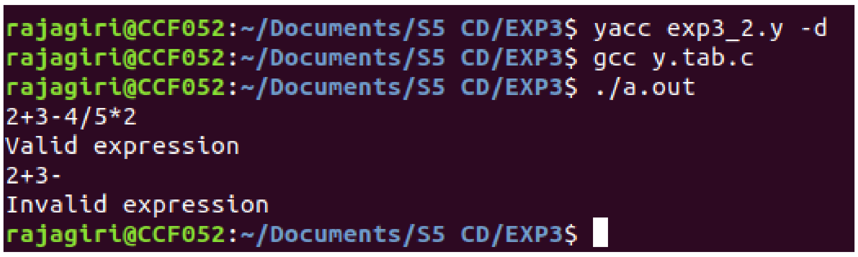
EXPT 3.2

CODE



```
%{
#include <stdio.h>
int yylex();
void yyerror(const char* s);
%}

%token NUMBER
%left '+' '-'
%left '*' '/'

%%

program: expression '\n' {
    printf("Valid expression\n");
}
        | '\n'
        ;

expression: NUMBER
            | expression '+' expression
            | expression '-' expression
            | expression '*' expression
            | expression '/' expression
            ;

%%

void yyerror(const char* s) {
    printf("Invalid expression\n");
}

int main() {
    yyparse();
    return 0;
}

int yylex() {
    int c = getchar();
    if (c == EOF) return 0;
    if (c == '+' || c == '-' || c == '*' || c == '/') {
        return c;
    }
    if (c >= '0' && c <= '9') {
        ungetc(c, stdin);
        scanf("%d", &yylval);
        return NUMBER;
    }
    return c;
}
```

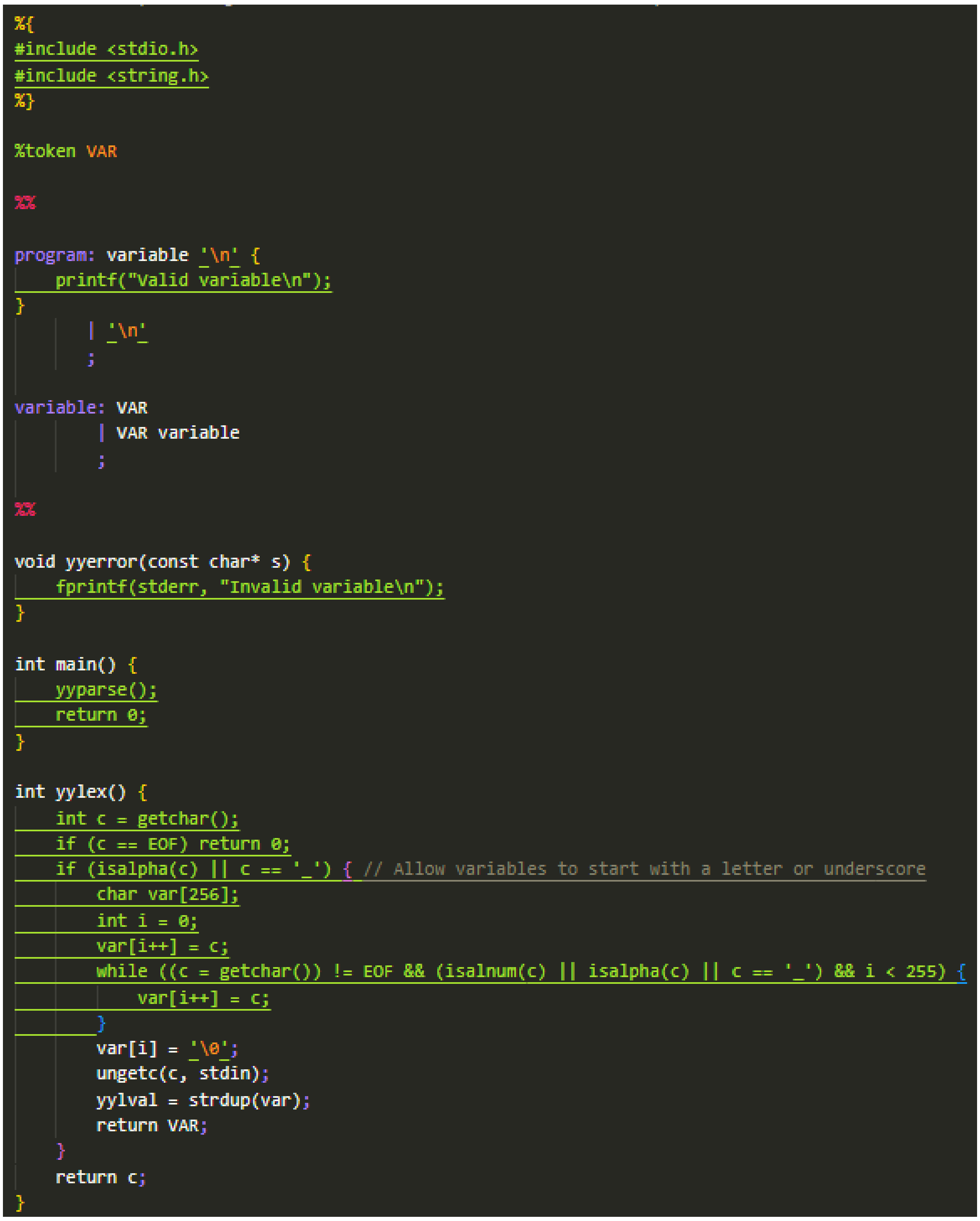
OUTPUT



```
rajagiri@CCF052:~/Documents/S5 CD/EXP3$ yacc exp3_2.y -d
rajagiri@CCF052:~/Documents/S5 CD/EXP3$ gcc y.tab.c
rajagiri@CCF052:~/Documents/S5 CD/EXP3$ ./a.out
2+3-4/5*2
Valid expression
2+3-
Invalid expression
rajagiri@CCF052:~/Documents/S5 CD/EXP3$ 
```

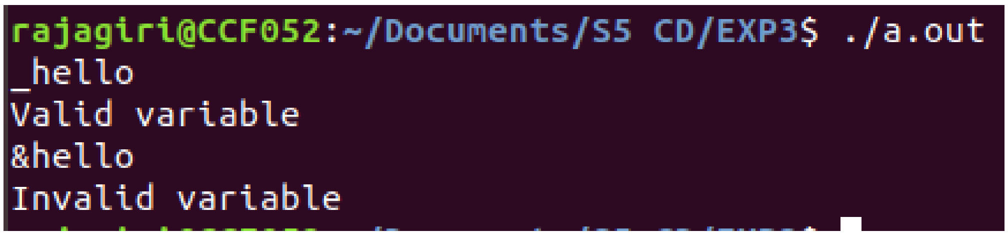
EXPT 3.3

CODE



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

%token VAR

%%

program: variable '\n' {
    printf("Valid variable\n");
}
       | '\n'
       ;

variable: VAR
        | VAR variable
        ;

%%

void yyerror(const char* s) {
    fprintf(stderr, "Invalid variable\n");
}

int main() {
    yyparse();
    return 0;
}

int yylex() {
    int c = getchar();
    if (c == EOF) return 0;
    if (isalpha(c) || c == '_') { // Allow variables to start with a letter or underscore
        char var[256];
        int i = 0;
        var[i++] = c;
        while ((c = getchar()) != EOF && (isalnum(c) || isalpha(c) || c == '_') && i < 255) {
            var[i++] = c;
        }
        var[i] = '\0';
        ungetc(c, stdin);
        yylval = strdup(var);
        return VAR;
    }
    return c;
}
```

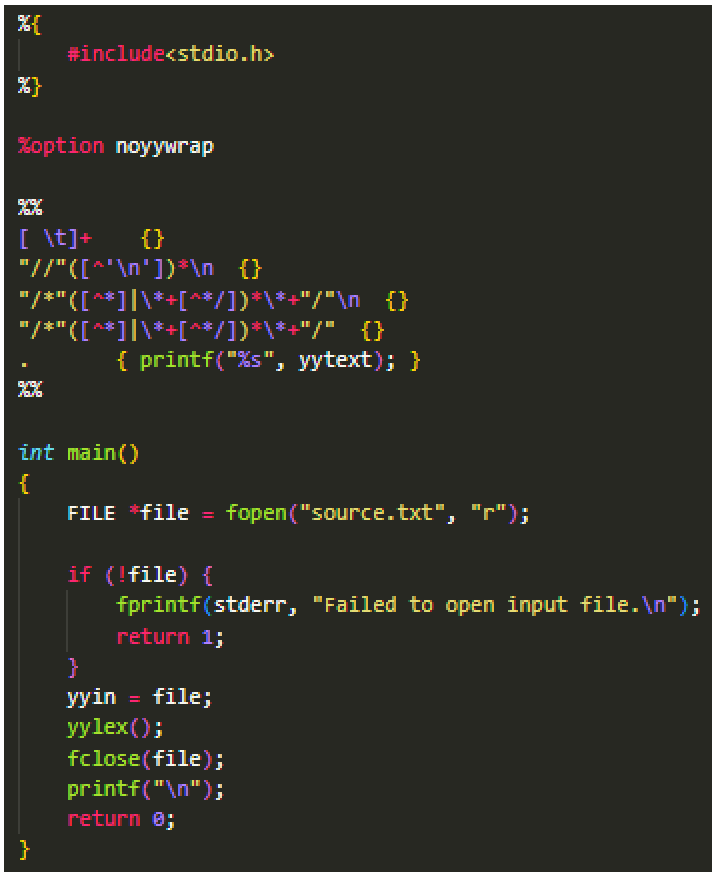
OUTPUT



```
rajagiri@CCF052:~/Documents/S5 CD/EXP3$ ./a.out
_hello
Valid variable
&hello
Invalid variable
```

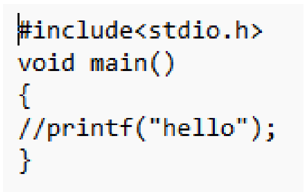
EXPT 4.1

CODE

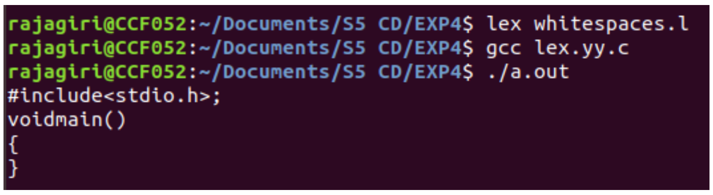


```
%{
    #include<stdio.h>
%}

%option noyywrap

%%
[ \t]+    {}
"//"([^'\n'])*\n  {}
"/*"([^*]|\*+[^*/])*\*+"/"\n  {}
"/*"([^*]|\*+[^*/])*\*+"/"  {}
.       { printf("%s", yytext); }
%%

int main()
{
    FILE *file = fopen("source.txt", "r");

    if (!file) {
        fprintf(stderr, "Failed to open input file.\n");
        return 1;
    }
    yyin = file;
    yylex();
    fclose(file);
    printf("\n");
    return 0;
}
```

OUTPUT



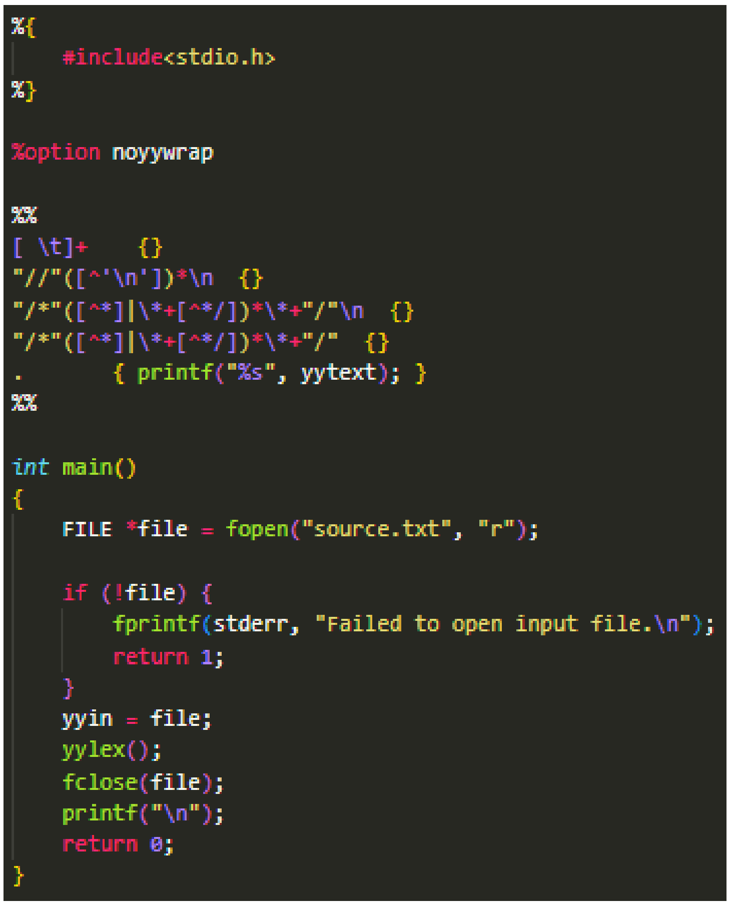
```
#include<stdio.h>
void main()
{
//printf("hello");
}
```



```
rajagiri@CCF052:~/Documents/S5 CD/EXP4$ lex whitespaces.l
rajagiri@CCF052:~/Documents/S5 CD/EXP4$ gcc lex.yy.c
rajagiri@CCF052:~/Documents/S5 CD/EXP4$ ./a.out
#include<stdio.h>;
voidmain()
{
}
```

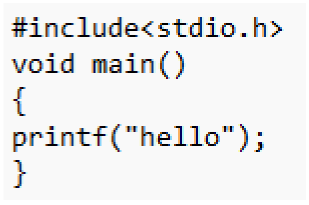
EXPT 4.2

CODE

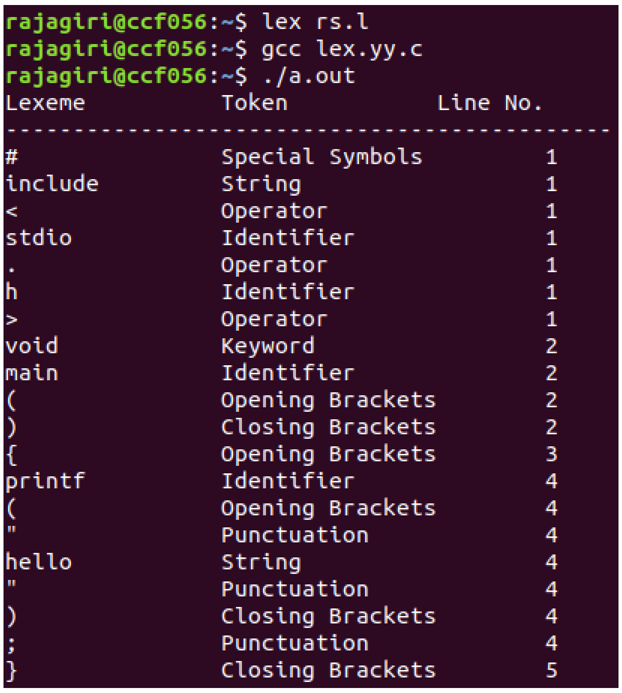


```
%{
    #include<stdio.h>
%}

%option noyywrap

%%
[ \t]+    {}
"//"([^'\n'])*\n  {}
"/*"([^*]|\*+[^*/])*\*+"/"\n  {}
"/*"([^*]|\*+[^*/])*\*+"/"  {}
.       { printf("%s", yytext); }
%%

int main()
{
    FILE *file = fopen("source.txt", "r");

    if (!file) {
        fprintf(stderr, "Failed to open input file.\n");
        return 1;
    }
    yyin = file;
    yylex();
    fclose(file);
    printf("\n");
    return 0;
}
```

OUTPUT



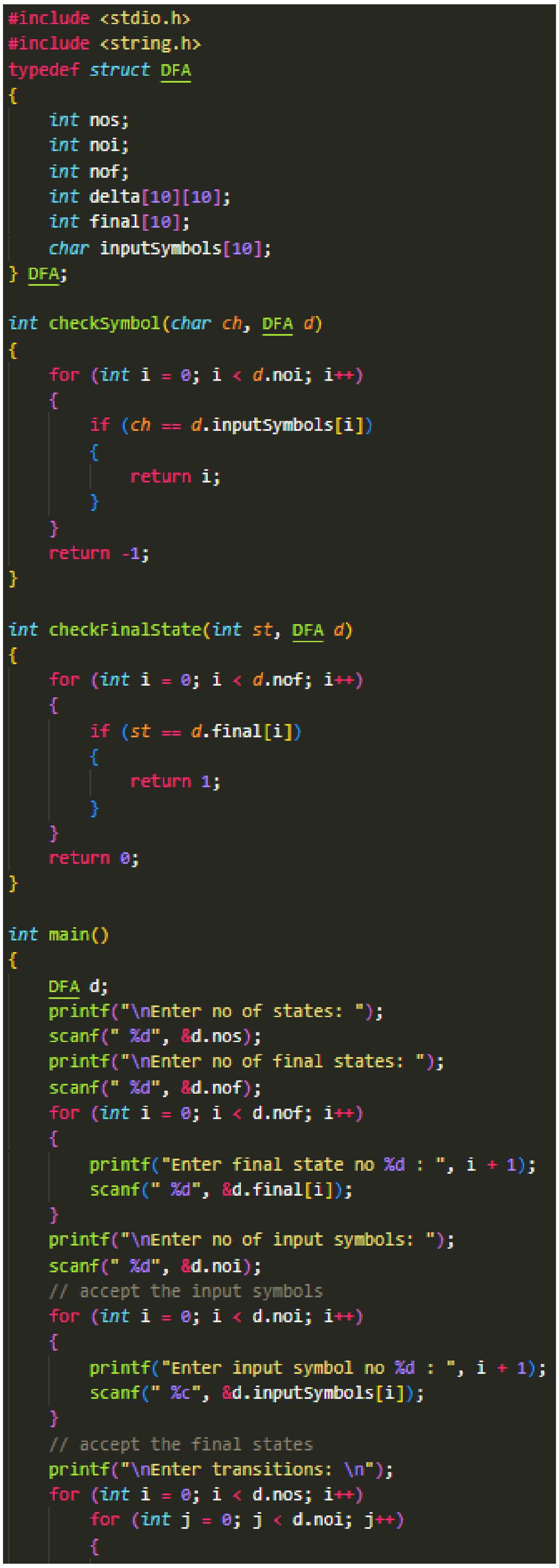
```
#include<stdio.h>
void main()
{
printf("hello");
}
```



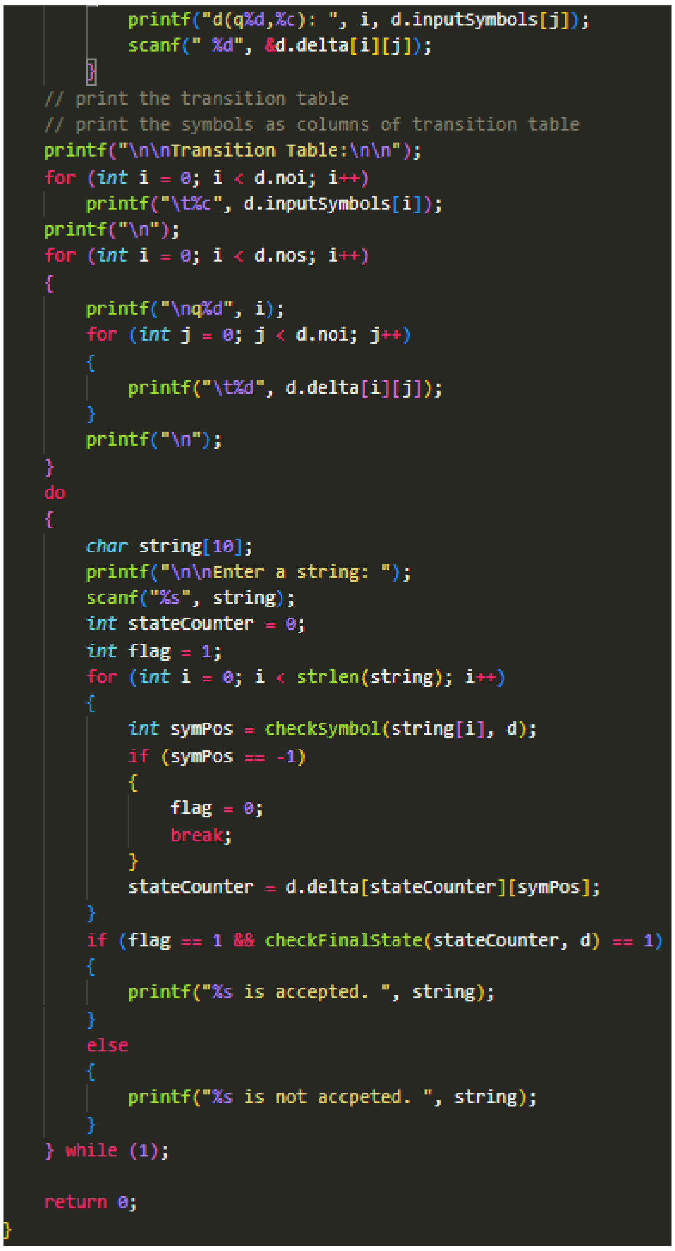
```
rajagiri@ccf056:~$ lex rs.l
rajagiri@ccf056:~$ gcc lex.yy.c
rajagiri@ccf056:~$ ./a.out
Lexeme          Token           Line No.
----------------------------------------------
#               Special Symbols         1
include         String                  1
<               Operator                1
stdio           Identifier              1
.               Operator                1
h               Identifier              1
>               Operator                1
void            Keyword                 2
main            Identifier              2
(               Opening Brackets        2
)               Closing Brackets        2
{               Opening Brackets        3
printf          Identifier              4
(               Opening Brackets        4
"               Punctuation             4
hello           String                  4
"               Punctuation             4
)               Closing Brackets        4
;               Punctuation             4
}               Closing Brackets        5
```

EXPT 5

CODE

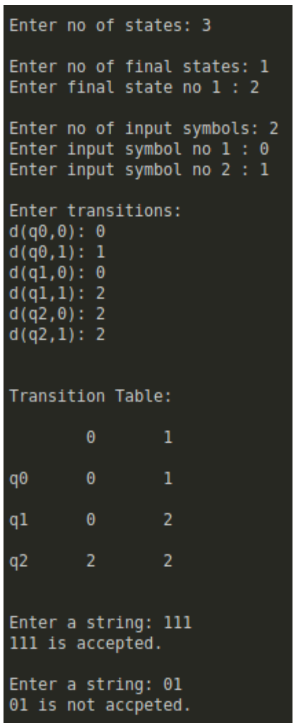


```c
#include <stdio.h>
#include <string.h>
typedef struct DFA
{
    int nos;
    int noi;
    int nof;
    int delta[10][10];
    int final[10];
    char inputSymbols[10];
} DFA;

int checkSymbol(char ch, DFA d)
{
    for (int i = 0; i < d.noi; i++)
    {
        if (ch == d.inputSymbols[i])
        {
            return i;
        }
    }
    return -1;
}

int checkFinalState(int st, DFA d)
{
    for (int i = 0; i < d.nof; i++)
    {
        if (st == d.final[i])
        {
            return 1;
        }
    }
    return 0;
}

int main()
{
    DFA d;
    printf("\nEnter no of states: ");
    scanf(" %d", &d.nos);
    printf("\nEnter no of final states: ");
    scanf(" %d", &d.nof);
    for (int i = 0; i < d.nof; i++)
    {
        printf("Enter final state no %d : ", i + 1);
        scanf(" %d", &d.final[i]);
    }
    printf("\nEnter no of input symbols: ");
    scanf(" %d", &d.noi);
    // accept the input symbols
    for (int i = 0; i < d.noi; i++)
    {
        printf("Enter input symbol no %d : ", i + 1);
        scanf(" %c", &d.inputSymbols[i]);
    }
    // accept the final states
    printf("\nEnter transitions: \n");
    for (int i = 0; i < d.nos; i++)
        for (int j = 0; j < d.noi; j++)
        {
```



```c
            printf("d(q%d,%c): ", i, d.inputSymbols[j]);
            scanf(" %d", &d.delta[i][j]);
        }
    // print the transition table
    // print the symbols as columns of transition table
    printf("\n\nTransition Table:\n\n");
    for (int i = 0; i < d.noi; i++)
        printf("\t%c", d.inputSymbols[i]);
    printf("\n");
    for (int i = 0; i < d.nos; i++)
    {
        printf("\nq%d", i);
        for (int j = 0; j < d.noi; j++)
        {
            printf("\t%d", d.delta[i][j]);
        }
        printf("\n");
    }
    do
    {
        char string[10];
        printf("\n\nEnter a string: ");
        scanf("%s", string);
        int stateCounter = 0;
        int flag = 1;
        for (int i = 0; i < strlen(string); i++)
        {
            int symPos = checkSymbol(string[i], d);
            if (symPos == -1)
            {
                flag = 0;
                break;
            }
            stateCounter = d.delta[stateCounter][symPos];
        }
        if (flag == 1 && checkFinalState(stateCounter, d) == 1)
        {
            printf("%s is accepted. ", string);
        }
        else
        {
            printf("%s is not accpeted. ", string);
        }
    } while (1);

    return 0;
}
```

-> OUTPUT



```
Enter no of states: 3

Enter no of final states: 1
Enter final state no 1 : 2

Enter no of input symbols: 2
Enter input symbol no 1 : 0
Enter input symbol no 2 : 1

Enter transitions:
d(q0,0): 0
d(q0,1): 1
d(q1,0): 0
d(q1,1): 2
d(q2,0): 2
d(q2,1): 2


Transition Table:

        0       1

q0      0       1

q1      0       2

q2      2       2


Enter a string: 111
111 is accepted.

Enter a string: 01
01 is not accpeted.
```