```
// #include "/home/student/Desktop/KaustavLABS3/CD LAB/LAB
04/lab04_q1_symbol_table_header.h"
#include "/home/kaustav/Desktop/KaustavLABS3/CD LAB/LAB
04/lab04_q1_symbol_table_header.h"

int curr = 0;
// char str[100];
static char str[700000000];

// FILE *fp = fopen("lab04_q1_input.c", "r");
FILE *fp;
struct token *currentToken;
////////////////////////////////////////////////////////////////////////////
////////////////////////////////

// LAB 07
void Program();
void declarations();
void data_type();
void identifier_list();
void identifier_list_factors();
void identifier_list_factors_array();
void assign_stat();
void assign_stat_factors();

// LAB 08
void statement_list();
void statement();
void expn();
void eprime();
void simple_expn();
void seprime();
void term();
void tprime();
void factor();
void relop();
void addop();
void mulop();

////////////////////////////////////////////////////////////////////////////
////////////////////////////////
void success()
{
    printf("SUCCESS\n");
    exit(0);
}

void invalid()
{
    printf("Error at Row %d : Column %d ::", currentToken->row, currentToken-
>column);
    exit(0);
}

void tokenDebug()
{
    printf("Token Scanned < %s , %s > \n ", currentToken->lexeme, currentToken-
>type);
    // insert_into_local_symbol_table_helper(currentToken);
}

////////////////////////////////////////////////////////////////////////////
////////////////////////////////
```

```
void Program()
{
    if (strcmp(currentToken->lexeme, "main") == 0)
    {
        currentToken = getNextToken(fp), tokenDebug();
        if (strcmp(currentToken->lexeme, "(") == 0)
        {
            currentToken = getNextToken(fp), tokenDebug();
            if (strcmp(currentToken->lexeme, ")") == 0)
            {
                currentToken = getNextToken(fp), tokenDebug();
                if (strcmp(currentToken->lexeme, "{") == 0)
                {
                    currentToken = getNextToken(fp), tokenDebug();

                    declarations();
                    statement_list();

                    if (strcmp(currentToken->lexeme, "}") == 0)
                        return;
                    else
                    {
                        printf("} expected \n");
                        invalid();
                    }
                }
                else
                {
                    printf("{ expected \n");
                    invalid();
                }
            }
            else
            {
                printf(") expected \n");
                invalid();
            }
        }
        else
        {
            printf("( expected \n");
            invalid();
        }
    }
    else
    {
        printf("main expected \n");
        invalid();
    }
}

void declarations()
{
    char first_of_declarations[2][10] = {"int", "char"};
    int flag = 0;
    for (int i = 0; i < sizeof(first_of_declarations) /
sizeof(first_of_declarations[0]); ++i)
    {
        if (strcmp(currentToken->lexeme, first_of_declarations[i]) == 0)
            flag++;
    }

    if (flag)
```

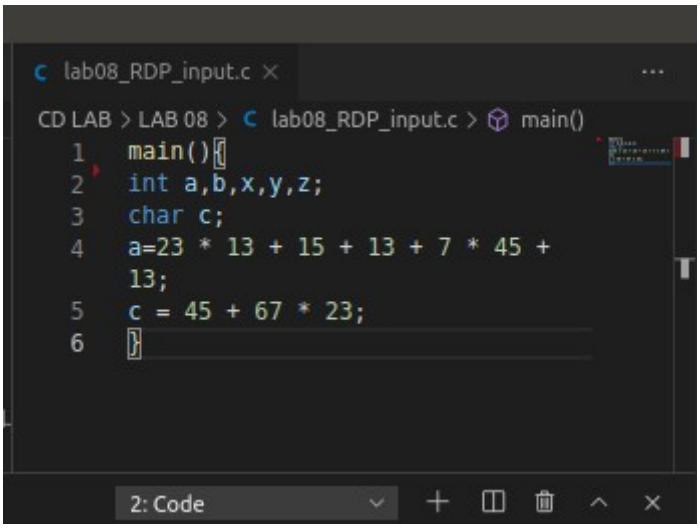
```
    {
        data_type();
        identifier_list();
        if (strcmp(currentToken->lexeme, ";") == 0)
        {
            currentToken = getNextToken(fp), tokenDebug();
            declarations();
        }
        else
        {
            printf("here ; expected \n");
            invalid();
        }
    }
}

void data_type()
{
    if ((strcmp(currentToken->lexeme, "int") == 0 || strcmp(currentToken-
>lexeme, "char") == 0))
    {
        currentToken = getNextToken(fp), tokenDebug();
        return;
    }
}

void identifier_list()
{
    if (strcmp(currentToken->type, "identifier") == 0)
    {
        currentToken = getNextToken(fp), tokenDebug();
        identifier_list_factors();
    }
    else
    {
        printf("identifier expected\n");
        invalid();
    }
}

void identifier_list_factors()
{
    if (strcmp(currentToken->lexeme, ",") == 0)
    {
        currentToken = getNextToken(fp), tokenDebug();
        identifier_list();
    }
    else if (strcmp(currentToken->lexeme, "[") == 0)
    {
        currentToken = getNextToken(fp), tokenDebug();
        if (strcmp(currentToken->type, "constant") == 0)
        {
            currentToken = getNextToken(fp), tokenDebug();
            if (strcmp(currentToken->lexeme, "]") == 0)
            {
                currentToken = getNextToken(fp), tokenDebug();
                identifier_list_factors_array();
            }
            else
            {
                printf("] expected \n");
                invalid();
            }
        }
```

```
        else
        {
            printf("constant expected \n");
            invalid();
        }
    }
    // else
    // {
    //     printf(", or [ expected \n");
    //     invalid();
    // }
}

void identifier_list_factors_array()
{
    if (strcmp(currentToken->lexeme, ",") == 0)
    {
        currentToken = getNextToken(fp), tokenDebug();
        identifier_list();
    }
}

void assign_stat()
{
    if (strcmp(currentToken->type, "identifier") == 0)
    {
        currentToken = getNextToken(fp), tokenDebug();
        if (strcmp(currentToken->lexeme, "=") == 0)
        {
            currentToken = getNextToken(fp), tokenDebug();
            // assign_stat_factors();
            expn();
        }
        else
        {
            printf("= expected\n");
            invalid();
        }
    }
    else
    {
        printf("identifier expected\\n");
        invalid();
    }
}

void statement_list()
{
    if (strcmp(currentToken->type, "identifier") == 0)
    {
        statement();
        statement_list();
    }
}
void statement()
{
    assign_stat();
    if (strcmp(currentToken->lexeme, ";") == 0)
    {
        currentToken = getNextToken(fp), tokenDebug();
        return;
    }
    else
    {
```

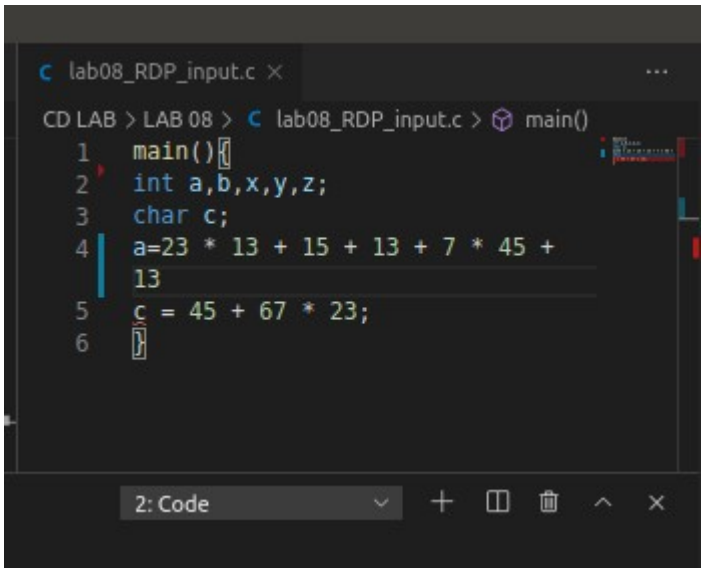
```
        printf("; expected \n");
        invalid();
    }
}
void expn()
{
    simple_expn();
    eprime();
}
void eprime()
{
    if (strcmp(currentToken->type, "relational_operators") == 0)
    {
        currentToken = getNextToken(fp), tokenDebug();
        simple_expn();
    }
}
void simple_expn()
{
    term();
    seprime();
}

void seprime()
{
    char first_of_seprime[2][2] = {"+", "-"};
    int flag = 0;

    for (int i = 0; i < sizeof(first_of_seprime) / sizeof(first_of_seprime[0]);
++i)
    {
        if (strcmp(currentToken->lexeme, first_of_seprime[i]) == 0)
            flag++;
    }

    if (flag)
    {
        addop();
        term();
        seprime();
    }
}

void term()
{
    factor();
    tprime();
}

void tprime()
{
    char first_of_tprime[3][3] = {"*", "/", "%"};
    int flag = 0;

    for (int i = 0; i < sizeof(first_of_tprime) / sizeof(first_of_tprime[0]); +
+i)
    {
        if (strcmp(currentToken->lexeme, first_of_tprime[i]) == 0)
            flag++;
    }

    if (flag)
    {
        mulop();
```

```
        factor();
        tprime();
    }
}
void factor()
{
    if (strcmp(currentToken->type, "identifier") == 0 || strcmp(currentToken-
>type, "constant") == 0)
    {
        currentToken = getNextToken(fp), tokenDebug();
        return;
    }
    else
    {
        printf("identifier or constant expected \n");
        invalid();
    }
}
void relop()
{
    if (strcmp(currentToken->type, "relational_operators") == 0)
    {
        currentToken = getNextToken(fp), tokenDebug();
        return;
    }
    else
    {
        printf("relational_operators expected \n");
        invalid();
    }
}

void addop()
{
    if (strcmp(currentToken->lexeme, "+") == 0 || strcmp(currentToken->lexeme,
"-") == 0)
    {
        currentToken = getNextToken(fp), tokenDebug();
        return;
    }
    else
    {
        printf("+ or - expected \n");
        invalid();
    }
}
void mulop()
{
    if (strcmp(currentToken->lexeme, "*") == 0 || strcmp(currentToken->lexeme,
"/") == 0 || strcmp(currentToken->lexeme, "%") == 0)
    {
        currentToken = getNextToken(fp), tokenDebug();
        return;
    }
    else
    {
        printf("* / or mod expected \n");
        invalid();
    }
}

int main(int argc, char const *argv[])
{
```

```
    fp = fopen("lab08_RDP_input.c", "r");
    // freopen("lab07_RDP_output.txt", "w", stdout);

    if (fp == NULL)
    {
        printf("Cannot open file \n Exiting.. \n");
        exit(0);
    }

    currentToken = getNextToken(fp), tokenDebug();
    Program();
    success();

    printf("\n******************Finished Recursive Decent
Parsing******************\n");

    return 0;
}
INPUT

main(){
int a,b,x,y,z;
char c;
a=23 * 13 + 15 + 13 + 7 * 45 + 13;
c = 45 + 67 * 23;
}
```




```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                                                                                                    2: Code

cd "/home/kaustav/Desktop/KaustavLABS3/CD LAB/LAB 08/" && gcc lab08_q1_RDP2.c -o lab08_q1_RDP2 && "/home/kaustav/Desktop/KaustavLABS3/CD LAB/LAB 08/"lab08_q1_RDP2
 kaustav@kaustav-OMEN-ubuntu  ~/Desktop/KaustavLABS3  main ±  cd "/home/kaustav/Desktop/KaustavLABS3/CD LAB/LAB 08/" && gcc lab08_q1_RDP2.c -o lab08_q1_RDP2 && "/home/kaustav/Desktop/KaustavLABS3/CD LAB
/LAB 08/"lab08_q1_RDP2
Token Scanned < main , function >
 Token Scanned < ( , special_symbols >
 Token Scanned < ) , special_symbols >
 Token Scanned < { , special_symbols >
 Token Scanned < int , keyword >
 Token Scanned < a , identifier >
 Token Scanned < , , special_symbols >
 Token Scanned < b , identifier >
 Token Scanned < , , special_symbols >
 Token Scanned < x , identifier >
 Token Scanned < , , special_symbols >
 Token Scanned < y , identifier >
 Token Scanned < , , special_symbols >
 Token Scanned < z , identifier >
 Token Scanned < ; , special_symbols >
 Token Scanned < char , keyword >
 Token Scanned < c , identifier >
 Token Scanned < ; , special_symbols >
 Token Scanned < a , identifier >
 Token Scanned < = , special_symbols >
 Token Scanned < 23 , constant >
 Token Scanned < * , special_symbols >
 Token Scanned < 13 , constant >
 Token Scanned < + , special_symbols >
 Token Scanned < 15 , constant >
 Token Scanned < + , special_symbols >
 Token Scanned < 13 , constant >
 Token Scanned < + , special_symbols >
 Token Scanned < 7 , constant >
 Token Scanned < * , special_symbols >
 Token Scanned < 45 , constant >
 Token Scanned < + , special_symbols >
 Token Scanned < 13 , constant >
 Token Scanned < ; , special_symbols >
 Token Scanned < c , identifier >
 Token Scanned < = , special_symbols >
 Token Scanned < 45 , constant >
 Token Scanned < + , special_symbols >
 Token Scanned < 67 , constant >
 Token Scanned < * , special_symbols >
 Token Scanned < 23 , constant >
 Token Scanned < ; , special_symbols >
 Token Scanned < } , special_symbols >
 SUCCESS
 kaustav@kaustav-OMEN-ubuntu  ~/Desktop/KaustavLABS3/CD LAB/LAB 08  main ±

Ln 358, Col 2  Spaces: 4  UTF-8  LF  c  Linux  kite: ready
```




```
kaustav@kaustav-OMEN-ubuntu ~/Desktop/KaustavLABS3/CD LAB/LAB 08 main ± cd "/home/kaustav/Desktop/KaustavLABS3/CD LAB/LAB 08/" && gcc lab08_q1_RDP2.c -o lab08_q1_RDP2 && "/home/kaustav/Desktop/Kaust
avLABS3/CD LAB/LAB 08/"lab08_q1_RDP2
Token Scanned < main , function >
 Token Scanned < ( , special_symbols >
 Token Scanned < ) , special_symbols >
 Token Scanned < { , special_symbols >
 Token Scanned < int , keyword >
 Token Scanned < a , identifier >
 Token Scanned < , , special_symbols >
 Token Scanned < b , identifier >
 Token Scanned < , , special_symbols >
 Token Scanned < x , identifier >
 Token Scanned < , , special_symbols >
 Token Scanned < y , identifier >
 Token Scanned < , , special_symbols >
 Token Scanned < z , identifier >
 Token Scanned < ; , special_symbols >
 Token Scanned < char , keyword >
 Token Scanned < c , identifier >
 Token Scanned < ; , special_symbols >
 Token Scanned < a , identifier >
 Token Scanned < = , special_symbols >
 Token Scanned < 23 , constant >
 Token Scanned < * , special_symbols >
 Token Scanned < 13 , constant >
 Token Scanned < + , special_symbols >
 Token Scanned < 15 , constant >
 Token Scanned < + , special_symbols >
 Token Scanned < 13 , constant >
 Token Scanned < + , special_symbols >
 Token Scanned < 7 , constant >
 Token Scanned < * , special_symbols >
 Token Scanned < 45 , constant >
 Token Scanned < + , special_symbols >
 Token Scanned < 13 , constant >
 Token Scanned < c , identifier >
 ; expected
Error at Row 4 : Column 31 ::
kaustav@kaustav-OMEN-ubuntu ~/Desktop/KaustavLABS3/CD LAB/LAB 08 main ±
```

Ln 357, Col 6 Spaces: 4 UTF-8 LF c Linux kite: ready