**System Software**

**Assignment 6**

**Name: Aman Kumar**

**Adm. No.: U19CS003**

**Q.**

**Write a program to implement Lexical Analyzer (Lexer).**

**CODE=>**

// U19CS003 AMAN KUMAR

#include <stdint.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

//STRUCTION TO STORE TOKENS

struct Token

{

    char name[1024];

    char type[128];

};

// GLOBAL ARRAY

struct Token tokens[2048];

int tk\_ct = 0;

// DELIMITERS FOR TOKENISING

int isdlim(char ch)

{

    if(ch == ' ')return 1;

    FILE \* fd=fopen("delimiter.txt","r");

    char c;

    while(fscanf(fd,"%c",&c)!=EOF)

    {

        if(ch==c)

        {

            fclose(fd);

            return 1;

        }

    }

    fclose(fd);

    return 0;

}

//FUNCTION TO CHECK IF STRING IS OPERATOR OR NOT

int isopr(char \*ch)

{

    if (strlen(ch) == 0)

        return 0;

    if (!strcmp(ch, "") || !strcmp(ch, "\n"))

        return 0;

    char \*c;

    FILE \*f = fopen("operator.txt", "r");

    char tmp[10];

    strcpy(tmp, ch);

    while (fscanf(f, "%s", c) != EOF)

    {

        if (!strcmp(tmp, c))

        {

            fclose(f);

            return 1;

        }

    }

    fclose(f);

    return 0;

}

//FUNCTION TO CHECK IF THE STRING IS PUNCTUATOR OR NOT

int ispunc(char \*ch)

{

    if (strlen(ch) == 0)

        return 0;

    if (!strcmp(ch, "") || !strcmp(ch, "\n"))

        return 0;

    char \*c;

    FILE \*f = fopen("punctuator.txt", "r");

    char tmp[10];

    strcpy(tmp, ch);

    while (fscanf(f, "%s", c) != EOF)

    {

        if (!strcmp(tmp, c))

        {

            fclose(f);

            return 1;

        }

    }

    fclose(f);

    return 0;

}

//FUNCTION TO CHECK IF THE STRING IS KEYWORD OR NOT

int iskey(char \*str)

{

    FILE \* fk=fopen("keywords.txt","r");

    char s[20];

    while(fscanf(fk,"%s",s)!=EOF)

    {

        if(!strcmp(str,s))

        {

            fclose(fk);

            return 1;

        }

    }

    fclose(fk);

    return 0;

}

//FUNCTION TO CHECK IF THE STRING IS VALID IDENTIFIER OR NOT

int viden(char \*str)

{

    if (strlen(str) == 0)

        return -1;

    if ((str[0] >= '0' && str[0] <= '9') || isdlim(str[0]) == 1 || iskey(str))

        return 0;

    if (str[0] == '\0' || str[0] == '\n')

        return -1;

    return 1;

}

//FUNCTION TO CHECK IF THE STRING IS AN INTEGER OR NOT

int isint(char \*str)

{

    int i, len = strlen(str);

    if (len == 0)

        return (0);

    for (i = 0; i < len; i++)

    {

        if (str[i] != '0' && str[i] != '1' && str[i] != '2' && str[i] != '3' && str[i] != '4' && str[i] != '5' && str[i] != '6' && str[i] != '7' && str[i] != '8' && str[i] != '9' || (str[i] == '-' && i > 0))

            return (0);

    }

    return (1);

}

//FUNCTION TO CHECK IF THE STRING IS DECIMAL OR NOT

int isdeci(char \*str)

{

    int i = 0;

    int flag = 0;

    while (str[i] != '\0')

    {

        if (str[i] != '0' && str[i] != '1' && str[i] != '2' && str[i] != '3' && str[i] != '4' && str[i] != '5' && str[i] != '6' && str[i] != '7' && str[i] != '8' && str[i] != '9' && str[i] != '.' || (str[i] == '-' && i > 0))

            return 0;

        if (str[i] == '.')

            flag = 1;

        i++;

    }

    return flag;

}

//FUNCTION TO CHECK IF THE STRING IS CONSTANT OR NOT

int is\_const(char \*str)

{

    if (isint(str))

        return 1;

    if (isdeci(str))

        return 1;

    if (strlen(str) < 3)

        return 0;

    if (str[0] == '\'' && str[strlen(str) - 1] == '\'')

    {

        return 1;

    }

}

//FUNCTION TO CHECK IF THE STRING IS STRING OR NOT

int is\_string(char \*str)

{

    if (strlen(str) < 3)

        return 0;

    // printf("%d\t%s",strlen(str),str);

    if (str[0] == '"' && str[strlen(str) - 1] == '"')

    {

        return 1;

    }

    return 0;

}

// FUNCTION TO EXTRACT SUBSTRING

char \*sbstr(char \*str, int l, int r)

{

    int i;

    char \*str1 = (char \*)malloc(sizeof(char) \* (r - l + 2));

    for (i = l; i <= r; i++)

        str1[i - l] = str[i];

    str1[r - l + 1] = '\0';

    return (str1);

}

// TOEKENIZING FUNCTION

void tokenise(char \*str)

{

    int l = 0, r = 0, len = strlen(str);

    int flag = 0;

    while (l <= r && r <= len)

    {

        // CHECKING FOR STRINGS

        // STRINGS ARE BOUNDED BY DOUBLE QUOTES

        if (str[r] == '"' && flag == 0)

        {

            flag = 1;

            r++;

        }

        if (flag)

        {

            if (str[r] == '"')

            {

                flag = 0;

                char \*str1 = sbstr(str, l, r);

                if (is\_string(str1) == 1)

                    strcpy(tokens[tk\_ct].name, str1);

                strcpy(tokens[tk\_ct++].type, "String");

            }

            r++;

            continue;

        }

        // IF NOT DELIMITER THEN CHECK FOR NEXT CHARACTER

        if (isdlim(str[r]) == 0)

            r++;

        // IF DELIMITER AND SINGLE CHARACTER

        if (isdlim(str[r]) == 1 && l == r)

        {

            // CHECKING IF THE CHARACTER IS PUNCTUATOR OR NOT

            if (str[r] != '.')

            {

                char \*st = sbstr(str, r, r);

                if (ispunc(st))

                {

                    strcpy(tokens[tk\_ct].name, st);

                    strcpy(tokens[tk\_ct++].type, "Punctuator");

                    r++;

                    l = r;

                    continue;

                }

            }

            // IF STRING IS '...' THEN ADD IT TO PUNCTUATOR

            else if (str[r + 1] == '.' && str[r + 2] == '.')

            {

                strcpy(tokens[tk\_ct].name, "...");

                strcpy(tokens[tk\_ct++].type, "Punctuator");

                r += 3;

                l = r;

                continue;

            }

            r++;

            // EXTARCT STRING OF CONTINUOUS OPERATORS

            while (isdlim(str[r]) && str[r] != ' ')

            {

                r++;

            }

            char \*str1 = sbstr(str, l, r - 1);

            // CHECK IF IT IS AN OPERATOR OR NOT

            if (isopr(str1) == 1)

            {

                strcpy(tokens[tk\_ct].name, str1);

                strcpy(tokens[tk\_ct++].type, "Operator");

            }

            l = r;

        }

        else if (isdlim(str[r]) == 1 && l != r || (r == len && l != r))

        {

            char \*str1 = sbstr(str, l, r - 1);

            // CHECK FOR KEYWORD

            if (iskey(str1) == 1)

            {

                strcpy(tokens[tk\_ct].name, str1);

                strcpy(tokens[tk\_ct++].type, "Keyword");

            }

            // CHECK FOR CONSTANTS

            else if (is\_const(str1) == 1)

            {

                strcpy(tokens[tk\_ct].name, str1);

                strcpy(tokens[tk\_ct++].type, "Constant");

            }

            // CONTINUE IF STRING AS ALREADY FOUND ABOVE

            else if (is\_string(str1) == 1)

            {

                l = r;

                continue;

            }

            // CHECK FOR VALID IDENTIFIER

            else if (viden(str1) == 1 && isdlim(str[r - 1]) == 0)

            {

                strcpy(tokens[tk\_ct].name, str1);

                strcpy(tokens[tk\_ct++].type, "Identifier");

            }

            l = r;

        }

    }

    return;

}

int main()

{

    char str[100];

    // OPENING INPUT FILE

    FILE \*f = fopen("input.txt", "r");

    int flag = 0, i;

    // READING LINE BY LINE

    while (fgets(str, 100, f))

    {

        // TO REMOVE COMMENTS

        flag = 0;

        int end = strlen(str);

        for (i = 0; i < strlen(str) - 1; i++)

        {

            if (str[i] == '/' && str[i + 1] == '/')

            {

                flag = 1;

                end = i - 1;

                break;

            }

        }

        if (flag)

        {

            char \*tmp;

            tmp = sbstr(str, 0, end);

            strcpy(str, tmp);

        }

        // TOKENISING AFTER REMOVING COMMENTS

        tokenise(str);

    }

    // CLOSING FILE

    fclose(f);

    // PRINTING TOKENS

    printf("\nToken Type          Token Name \n");

    char type[6][100] = {"Operator", "Punctuator", "Keyword", "Identifier", "Constant", "String"};

    for (i = 0; i < 6; i++)

    {

        int j = strlen(type[i]);

        printf("\n%s", type[i]);

        while (j < 15)

        {

            printf(" ");

            j++;

        }

        printf(":   ");

        for (j = 0; j < tk\_ct; j++)

        {

            if (!strcmp(tokens[j].type, type[i]))

                printf(" [%s],", tokens[j].name);

        }

    }

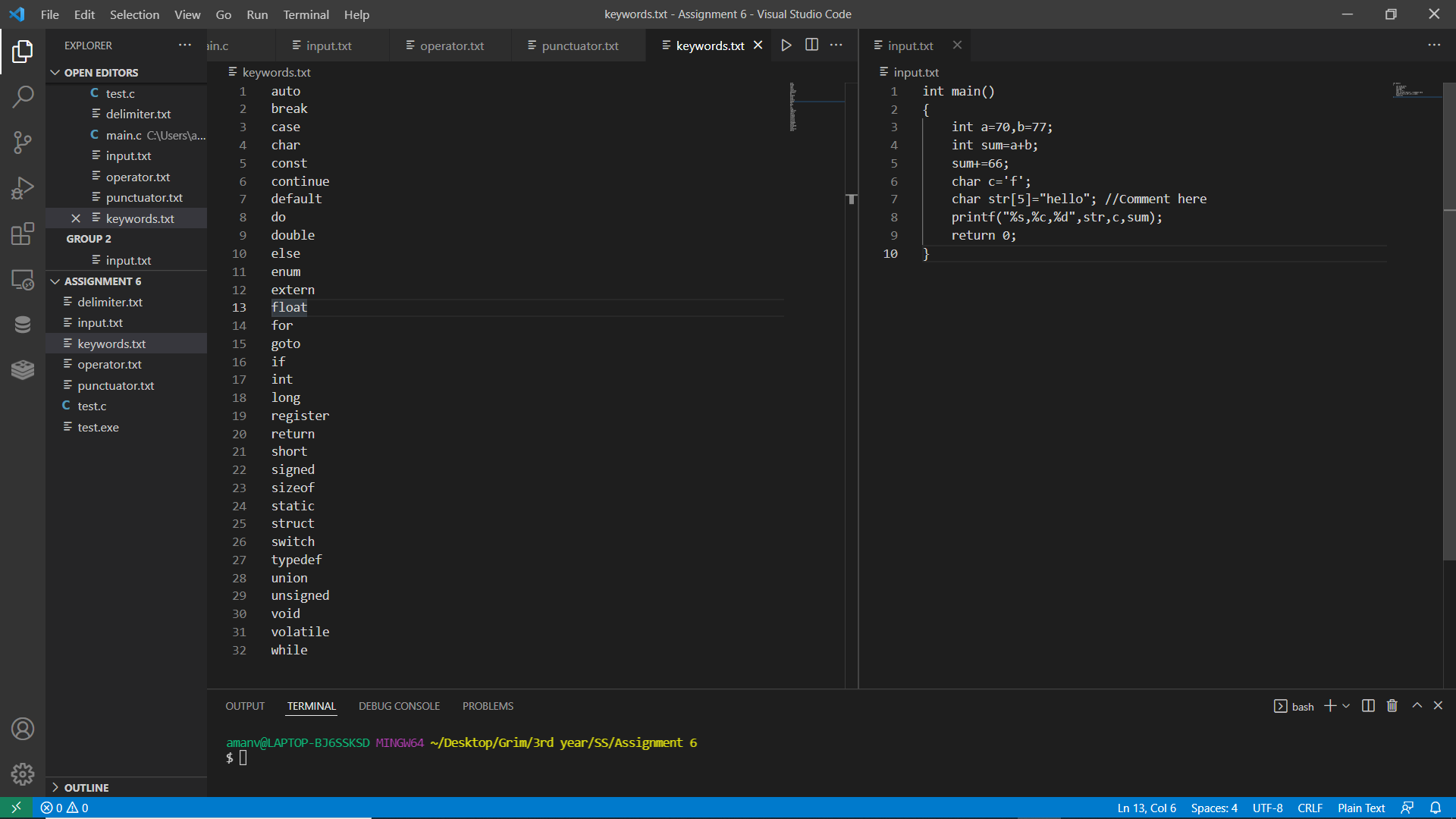
    printf("\n\nTotal Number of Tokens : %d\n", tk\_ct);

    return 0;

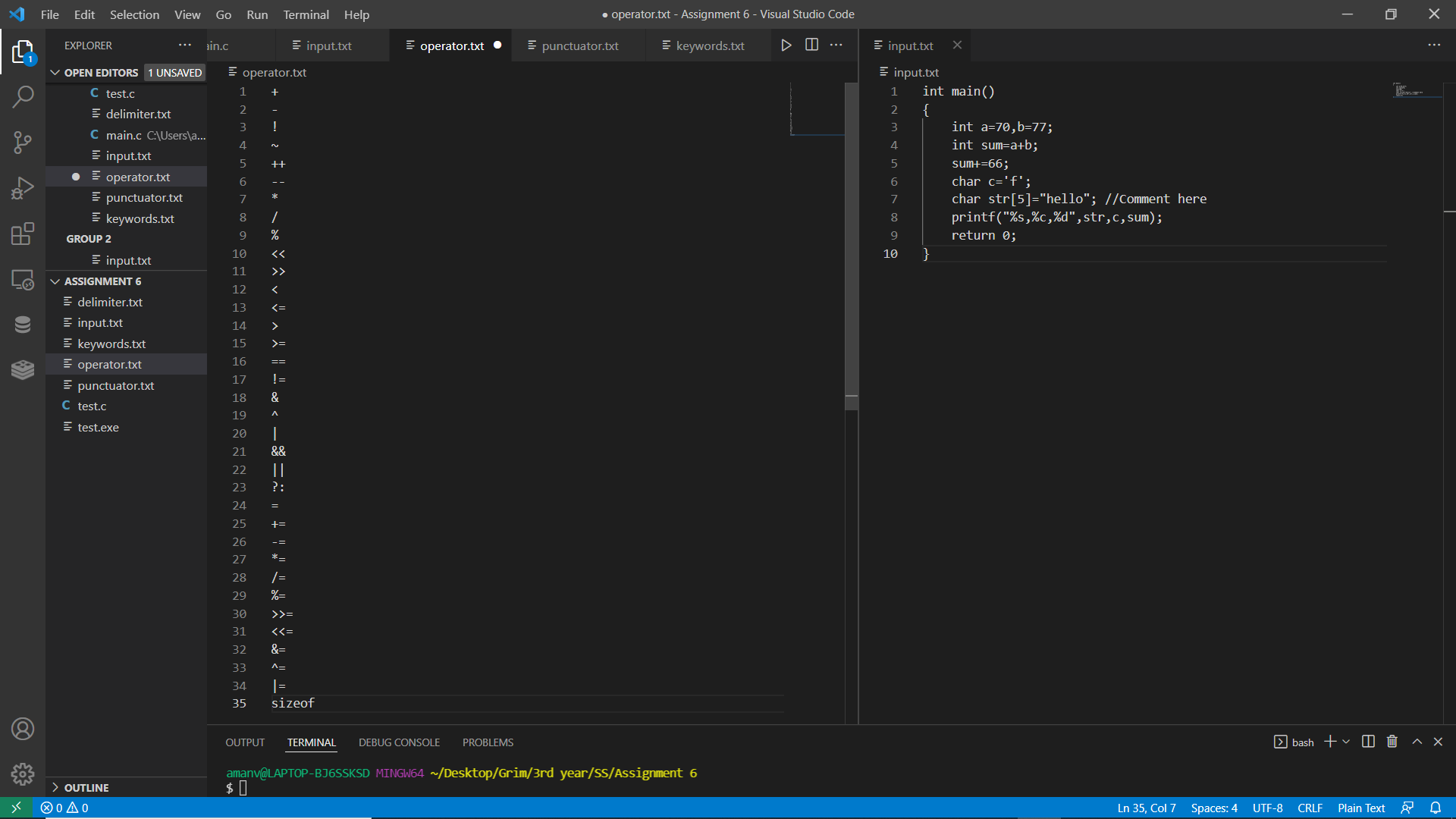
}

**FILES REQUIRED=>**

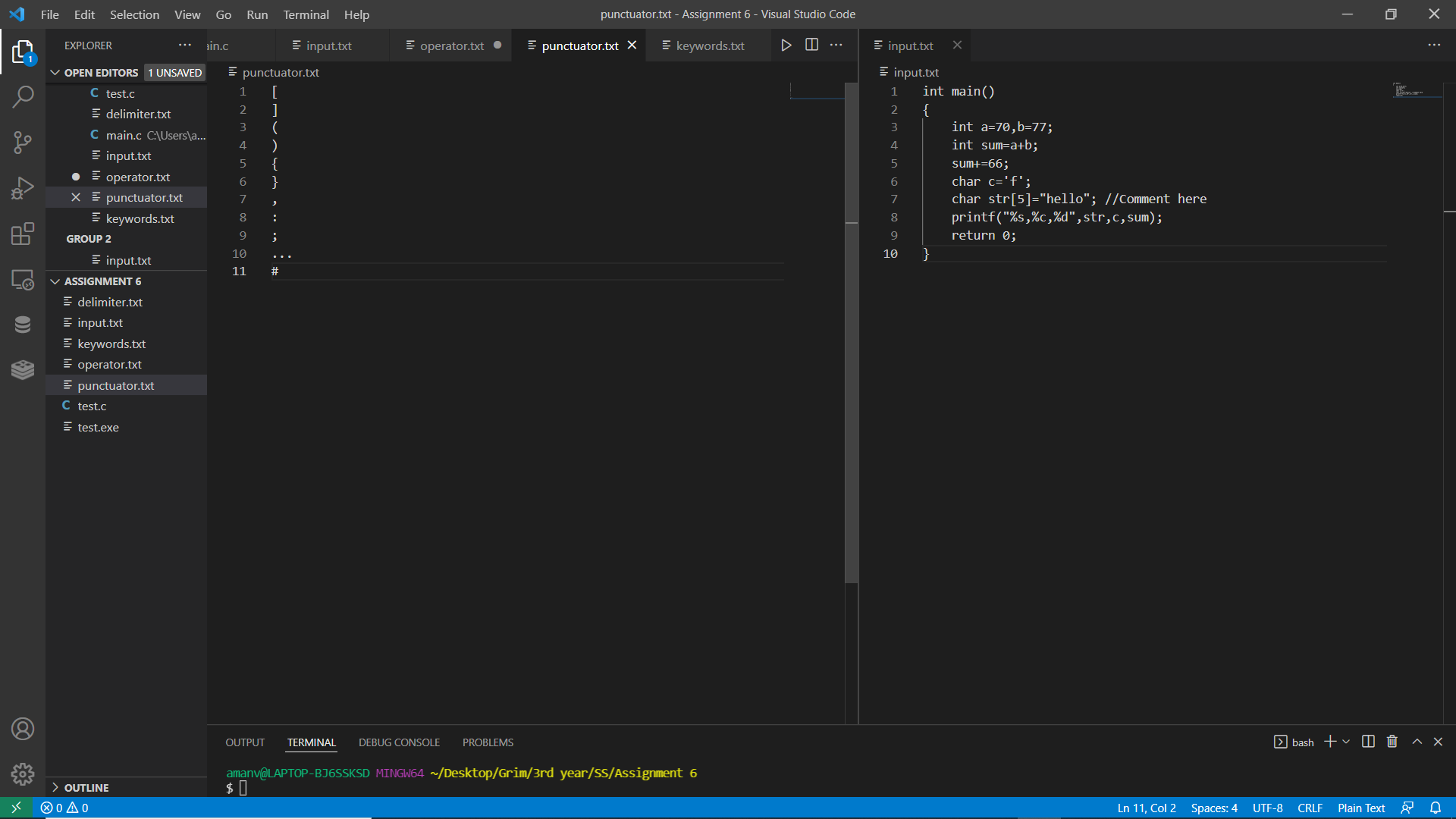
**Keywords.txt: Stores all 32 keywords of C language.**



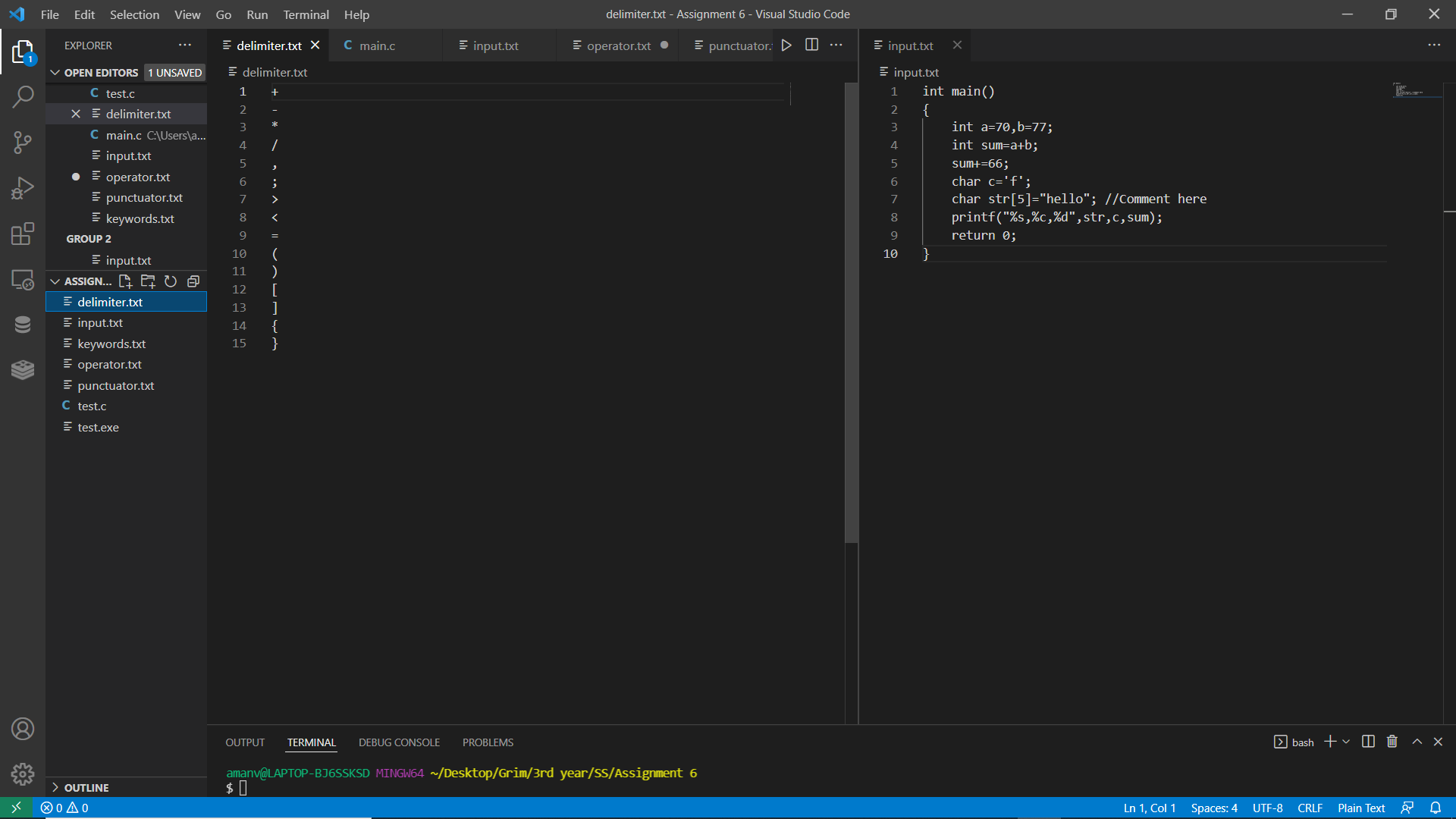
**Operator.txt: Stores all Operators**



**Punctuator.txt: Stores all Punctuators**

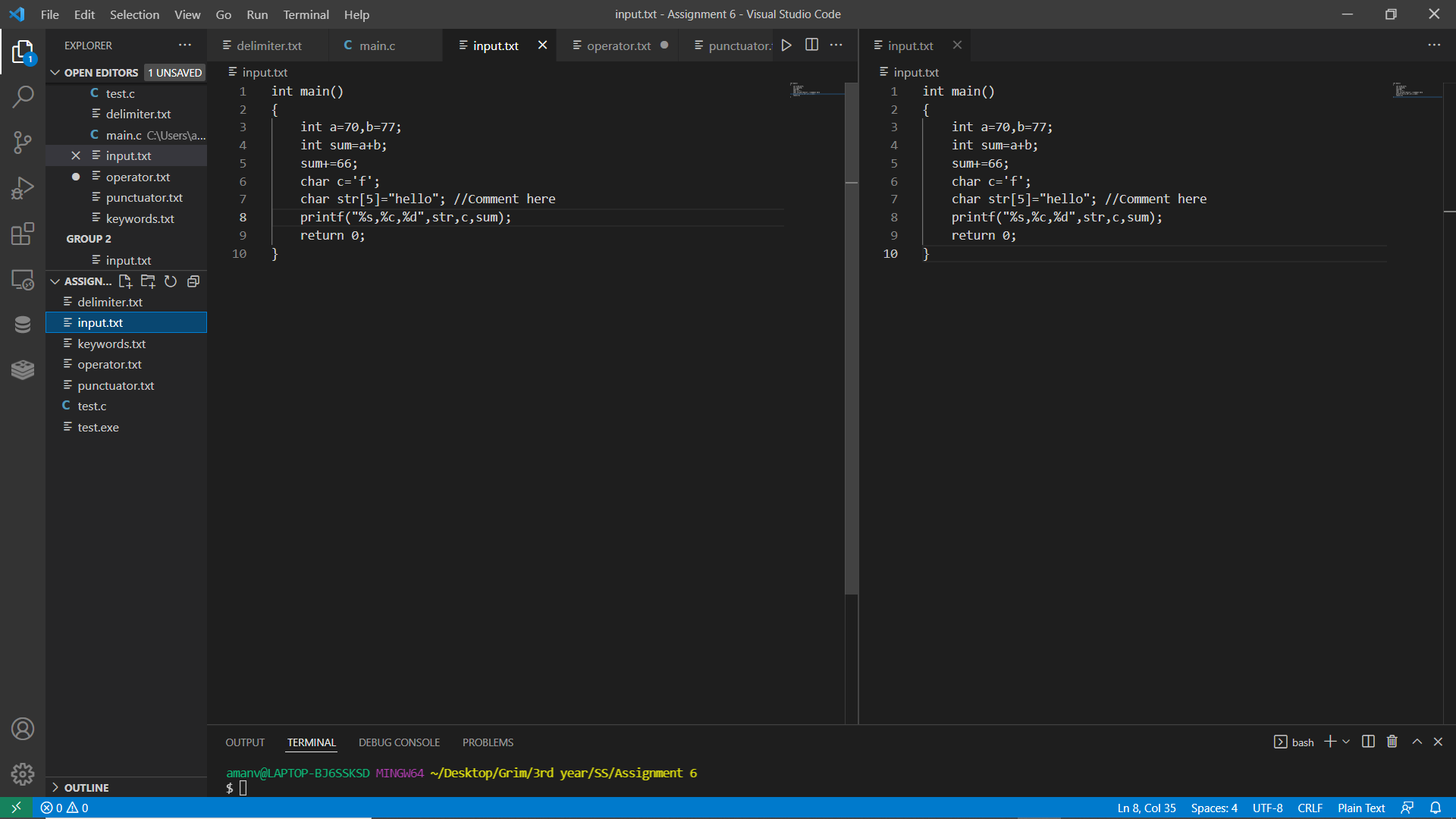


**Delimiter.txt: Stores all Delimiters**

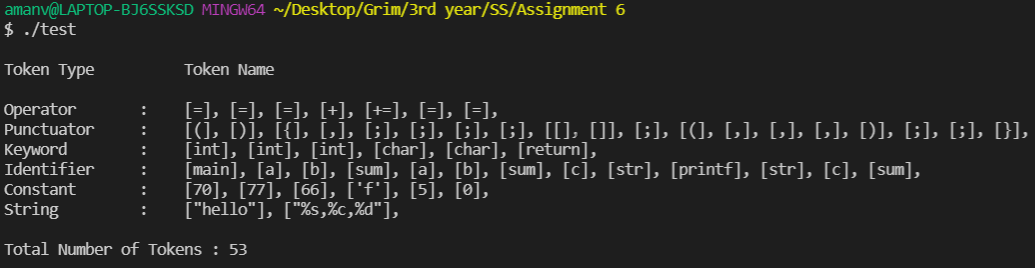


**INPUT=>**

**Input.txt: The code in C language to given as input to Lexer.**



**OUTPUT=>**

****