**Week 5 Assignment**

**Implement a Recursive Descent Parser for the Expression Grammar given below.**

**E → TE’**

**E’→ +TE’ | ͼ**

**T → FT’**

**T’→ \*FT’ | ͼ**

**F → (E) | i**

**Identify the grammar is suffers with left recursion and ambiguity before constructing the RD parser**

**Solution:**

The givne grammar doesn’t suffer with left recursion and ambiguity

**Program:**

#include <iostream>

#include <string.h>

using namespace std;

class Parser

{

private:

    string input;

    char \*input\_ptr;

public:

    Parser()

    {

    }

    Parser(string str)

    {

        input = str;

        input\_ptr = &input[0];

    }

    bool match(char required)

    {

        if (\*input\_ptr == required)

        {

            input\_ptr++;

            return true;

        }

        else

        {

            return false;

        }

    }

    bool F()

    {

        if (\*input\_ptr == '(')

        {

            if (match('(') && Start() && match(')'))

            {

                return true;

            }

            else

            {

                return false;

            }

        }

        else if (\*input\_ptr == 'i')

        {

            if (match('i'))

            {

                return true;

            }

            else

            {

                return false;

            }

        }

        return false;

    }

    bool T1()

    {

        if (\*input\_ptr == '\*')

        {

            if (match('\*') && F() && T1())

            {

                return true;

            }

            else

            {

                return false;

            }

        }

        return true;

    }

    bool T()

    {

        if (F() && T1())

        {

            return true;

        }

        return false;

    }

    bool E1()

    {

        if (\*input\_ptr == '+')

        {

            if (match('+') && T() && E1())

            {

                return true;

            }

            return false;

        }

        return true;

    }

    bool Start()

    {

        if (T() && E1())

        {

            return true;

        }

        else

        {

            return false;

        }

    }

    char get\_input\_ptr()

    {

        return \*input\_ptr;

    }

};

int main()

{

    string input;

    cout << "Enter the string: ";

    cin >> input;

    Parser Rd\_Parser(input);

    if (Rd\_Parser.Start() && Rd\_Parser.get\_input\_ptr() == '\0')

    {

        cout << "SYNTACTICALLY CORRECT" << endl;

    }

    else

    {

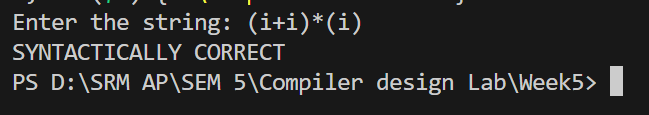
        cout << "SYNTACTICALLY INCORRECT";

    }

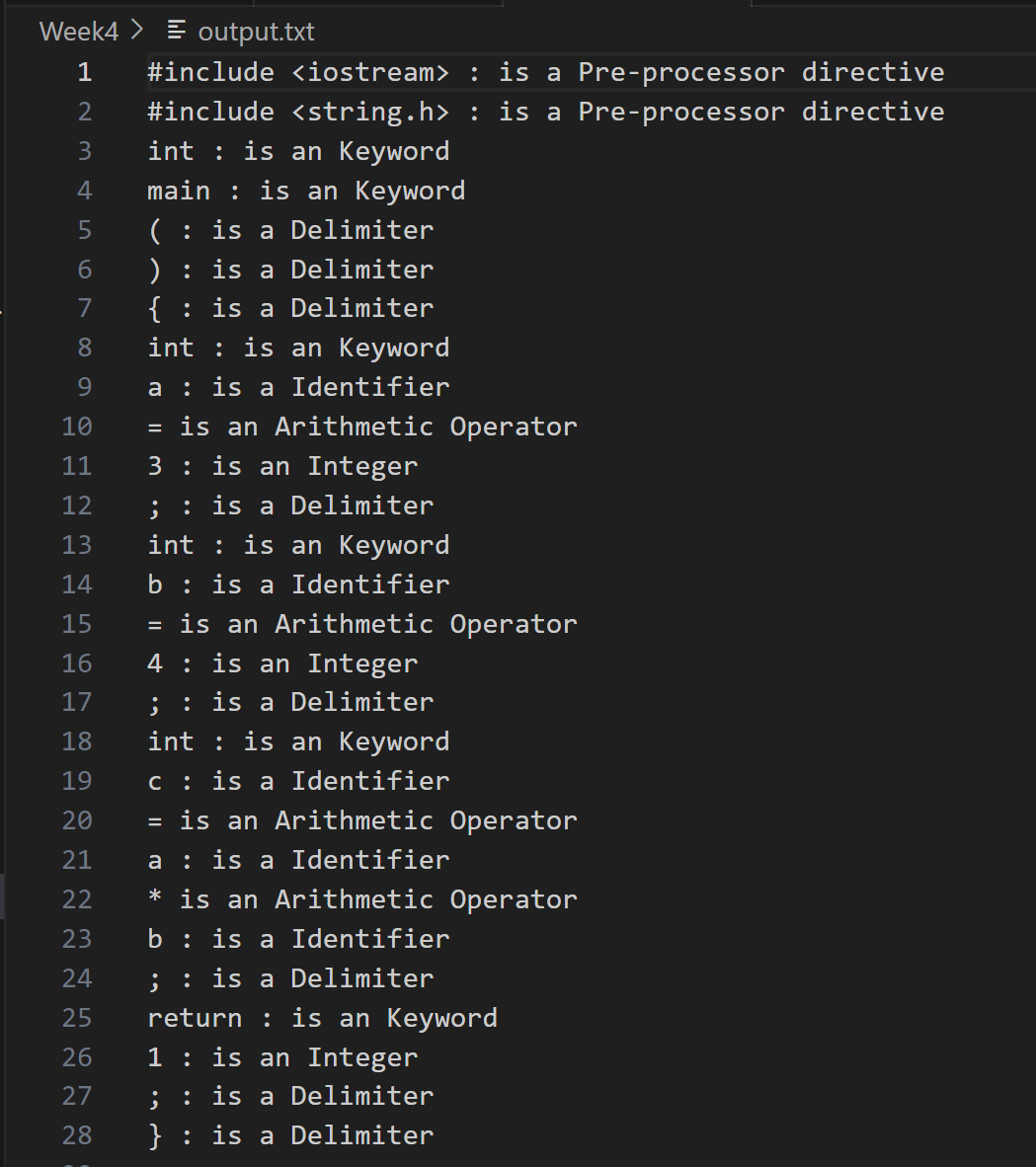
    return 1;

}

**Output:**

****

**Output file (.txt):**

****

**2. Write a C Program to Scan and Count the number of characters, words, and lines in a file.**

**Program:**

#include <stdio.h>

int main()

{

char filename[100];

printf("Enter the name of the file: ");

scanf("%s", filename);

FILE \*file = fopen(filename, "r");

if (file == NULL)

{

printf("Unable to open the file. Exiting...\n");

return 1;

}

int no\_of\_chars = 0;

int no\_of\_words = 0;

int no\_of\_lines = 0;

int in\_word = 0;

char ch;

while ((ch = fgetc(file)) != EOF)

{

no\_of\_chars++;

if (ch != ' ' && ch != '\t' && ch != '\n' && ch != '\r' && ch != '\f' && ch != '\v')

{

in\_word = 1;

}

if ((ch == ' ' || ch == '\t' || ch == '\n' || ch == '\r' || ch == '\f' || ch == '\v') &&

in\_word)

{

no\_of\_words++;

in\_word = 0;

}

if (ch == '\n' || ch == '\0')

{

no\_of\_lines++;

}

}

fclose(file);

printf("Number of characters: %d\n", no\_of\_chars);

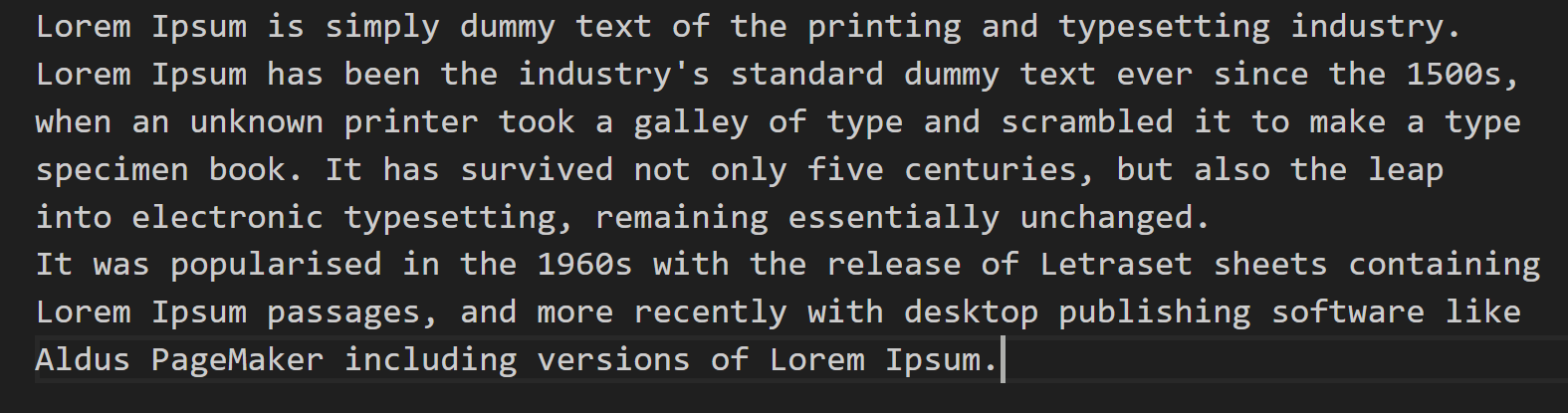
printf("Number of words: %d\n", no\_of\_words);

printf("Number of lines: %d\n", no\_of\_lines);

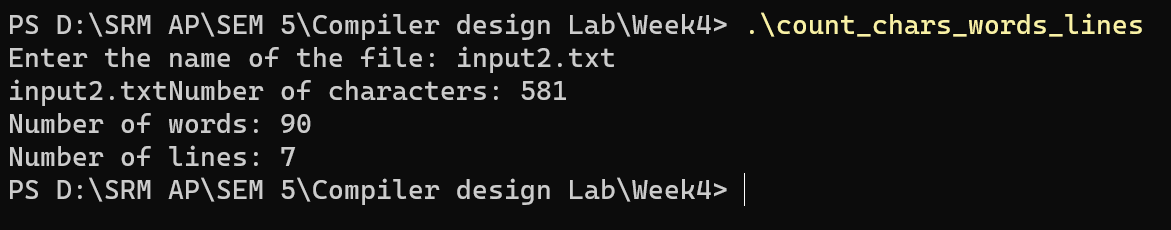
return 0;

}

**Input:**

****

**Ouptut:**

****