DEPARTMENT OF COMPUTER ENGINEERING

Semester	T.E. Semester VI– SPCC
Subject	Software Engineering
Subject Professor In-charge	Prof. Pankaj Vanvari
Assisting Teachers	Prof. Pankaj Vanvari
Laboratory	M310B

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

Title:

Tokenizer

Approach:

1. Read Input: The program reads input from a file, tokenizes it, and stores it in a vector for processing.
2. Tokenization: The input text is tokenized using a DFA approach. Each token is identified based on its type (identifier, integer literal, float literal, operator, etc.). Tokens are stored in a 2D vector along with their corresponding token ID.
3. Symbol/Literal Table: The program maintains separate symbol/literal tables for identifiers, integer literals, and float literals. Each entry in these tables is associated with a pointer or identifier.
4. Output: The program prints the tokens identified along with their token IDs and also displays the entries in the symbol/literal tables.

Implementation:

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <map>
using namespace std;
```

```
int readfile(string &fileName, vector<string> &input)
{
    char ch;
    fstream fp;
    fp.open(fileName.c_str(), std::fstream::in);

    if (!fp)
    {
        cerr << "Error opening the file: " << fileName << endl;
        return 1; // Return an error code
    }

    string word;
    while (fp >> noskipws >> ch)
    {
        if (ch == '\n')
        {
            input.push_back(word);
            input.push_back(";");
            word = "";
        }
        else if (ch == ' ')
        {
            input.push_back(word);
            word = "";
        }
        else
        {
            word += ch;
        }
    }

    input.push_back(word);

    fp.close();
    return 0; // Return success code
}

void print_vector_2D(vector<vector<string>> &input)
{
    for (int i = 0; i < input.size(); i++)
    {
        cout << input[i][0] << " " <<"*->"<< input[i][1] << " ";
```

```
        cout << endl;
    }
    cout << endl;
}

void print_vector(vector<string> &input)
{
    for (int i = 0; i < input.size(); i++)
    {
        cout << input[i] << " ";
    }
    cout << endl;
}

void Tokenization(vector<string> &input, vector<vector<string>> &Tokensed,
map<string, string> &keywords, map<string, int> &intcp, map<string, float>
&floatcp, map<string, string> &idp)
{
    int idc = 0;
    int intcc = 0;
    int floatcc = 0;

    for (int i = 0; i < input.size(); i++)
    {
        if (input[i] == ";")
            continue;

        if (keywords.find(input[i]) != keywords.end())
        {

            Tokensed.push_back({keywords[input[i]], "NA"});
        }
        else
        {
            // if the value in not in keyword db it can be eithre identifier or
constant

            string curr = input[i];
            char first_of_curr = curr[0]; // foc
            int val_of_foc = first_of_curr - '0';
            // cout << val_of_foc << endl;
            if (val_of_foc >= 0 && val_of_foc <= 9)
            {
                bool isfloat = false;
```

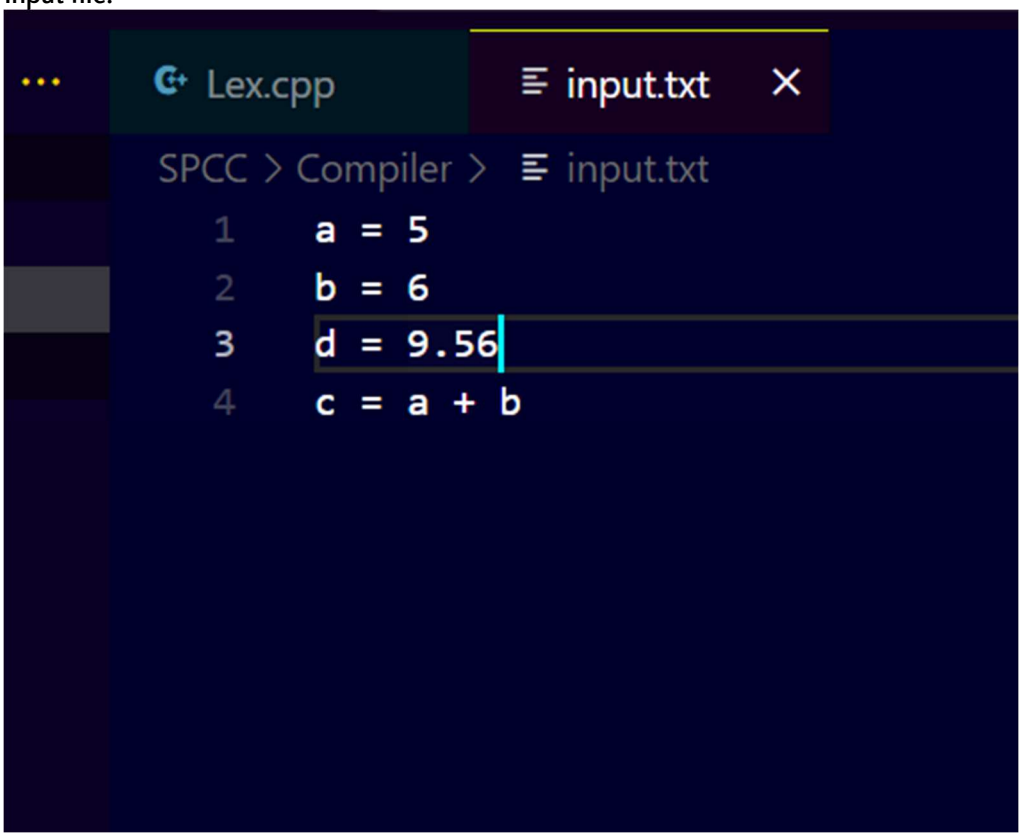
```
                for (auto x : curr)
                {
                    if (x == '.')
                        isfloat = true;
                }
                if (isfloat)
                {
                    float v = atof(curr.c_str());
                    string p = to_string(floatcc);
                    Tokensed.push_back({"3", p});
                    floatcp[p] = v;
                    floatcc++;
                }
                else
                {
                    int v = stoi(curr);
                    string p = to_string(intcc);
                    Tokensed.push_back({"2", p});
                    intcp[p] = v;
                    intcc++;
                }
            }
            else
            {
                string p = to_string(idc);

                Tokensed.push_back({"1", p});
                idp[p] = curr;
                idc++;
            }
        }
    }
}

void print_all_Symtabs(map<string, int> &intcp, map<string, float> &floatcp,
map<string, string> &idp)
{
    cout << "The integer constant pointer is: " << endl;
    for (auto x : intcp)
    {
        cout << x.first << "->" << x.second << endl;
    }
    cout << endl;
```

```
    cout << "The float constant pointer is: " << endl;
    for (auto x : floatcp)
    {
        cout << x.first << "->" << x.second << endl;
    }
    cout << endl;

    cout << "The identifier pointer is: " << endl;
    for (auto x : idp)
    {
        cout << x.first << "->" << x.second << endl;
    }
    cout << endl;
}

int main()
{
    // Database starts
    map<string, string> keywords;
    keywords["int"] = "INT";
    keywords["float"] = "FLOAT";
    keywords["+"] = "4";
    keywords["-"] = "5";
    keywords["*"] = "6";
    keywords["/"] = "7";
    keywords["="] = "9";
    keywords["^"] = "8";
    keywords["("] = "10";
    keywords[")"] = "11";

    // 1 for identifiers

    // pointer to intc
    map<string, int> intcp;
    // pointer ot intf
    map<string, float> floatcp;
    // pointer to identifier
    map<string, string> idp;



    // Database ends
    string inputFile;
    vector<string> input;
```

```
    vector<vector<string>> Tokensed;



    cout << "Enter the name of the file: ";
    cin >> inputFile;
    readfile(inputFile, input);

    cout << "The input file is: " << endl;
    print_vector(input);

    Tokenization(input, Tokensed, keywords, intcp, floatcp, idp);
    cout << "The tokens are: " << endl;
    print_vector_2D(Tokensed);

    print_all_Symtabs(intcp, floatcp, idp);



    return 0;
}
```

End Result:

Input file:



Output :

```
PROBLEMS    OUTPUT    TERMINAL    AZURE    PORTS    DEBUG CONSOLE

PS E:\GIt\SEM-6> cd "e:\GIt\SEM-6\SPCC\Compiler\" ; if ($?) { g++ Lex.cpp -o Lex } ; if ($?) { .\Lex }
Enter the name of the file: input.txt
The input file is:
a = 5 ; b = 6 ; d = 9.56 ; c = a + b
The tokens are:
1 *->0
9 *->NA
2 *->0
1 *->1
9 *->NA
2 *->1
1 *->2
9 *->NA
3 *->0
1 *->3
9 *->NA
1 *->4
4 *->NA
1 *->5

The integer constant pointer is:
0->5
1->6

The float constant pointer is:
0->9.56

The identifier pointer is:
0->a
1->b
2->d
3->c
4->a
5->b

PS E:\GIt\SEM-6\SPCC\Compiler>
```