COMSATS University Islamabad (CUI)

Lab Terminal

For

CC

By

Iqra Basharat(FA20-BCS-007)

Submitted To:

Sir Bilal

Bachelor of Science in Computer Science (2020-2024)

Question:01

2 functionailties along with input code and output.

Input Code:

```
using System;
using System.Collections.Generic;
namespace Analyzer {
    class Analyze {
        List<List<int>> rules = new List<List<int>>();
        private void loadTransitionTable(string path) {
            string text = System.IO.File.ReadAllText(path);
            if(text.Length < 2) {
                throw new Exception();
            }
            foreach(var item in text.Split('\n')) {
                var temp = new List<int>();
                foreach(var itm in item.Trim().Split(' ')) {
                    temp.Add(Convert.ToInt32(itm));
                }
                rules.Add(temp);
            }
        }

        private int getNextState(int iState,char cChar) {
            if(char.IsLetter(cChar))
                return rules[iState][1];
            else if(char.IsDigit(cChar))
                return rules[iState][2];
            else if(cChar == '.')
                return rules[iState][3];
            else if(cChar == '"')
                return rules[iState][4];
            else if(cChar == '\'')
                return rules[iState][5];
            else if(cChar == '_')
                return rules[iState][6];
            else if(cChar == '+')
                return rules[iState][7];
            else if(cChar == '=')
                return rules[iState][8];
            else if(cChar == '-')
                return rules[iState][9];
```

```
            else if(cChar == '%')
                return rules[iState][10];
            else if(cChar == '!')
                return rules[iState][11];
            else if(cChar == '>')
                return rules[iState][12];
            else if(cChar == '<')
                return rules[iState][13];
            else if(cChar == '/')
                return rules[iState][14];
            return rules[iState][0];
        }

        private bool isKeyword(string sToken) {
            if((sToken).Length > 16 || (sToken).Length == 0)
                return false;
            var sKeywords = new List<string>(){

"using","import","include","asm","auto","bool","break","case","catch","char","class","con
st","const_cast",

"continue","default","delete","do","double","dynamic_cast","else","enum","explicit",

"export","extern","false","float","for","friend","goto","if","inline","int","long",

"main","mutable","namespace","new","operator","private","protected","public",
                "register","reinterpret_cast","return","short","signed","sizeof","static",

"static_cast","struct","switch","template","this","throw","true","try","typedef",

"typeid","typename","union","unsigned","using","virtual","void","volatile","wchar_t","whi
le"};
            return sKeywords.Exists(element => (sToken.ToLower()) == element);
        }
        public string Result(string txt,string tt = @"matrix.txt") {
            try {
                loadTransitionTable(tt);
            }
            catch(Exception) {
                return "Unable to open the input file.\nPress any key to exit.\n";
            }
            if(txt.Length == 0)
                return "";
            var result = "";
            int txtIndex = 0,iState=0;
            char cTemp = txt[txtIndex], cChar = ' ';
            string sToken = "";
            bool flag = true;
            ///////
            txt += " ";
            while(txtIndex != txt.Length) {
                if(flag) {
                    cChar = cTemp;
                    if(txt.Length - 1 == txtIndex)
```

```
                return result + cChar;
            cTemp = txt[++txtIndex];
        }
        else
           flag = true;
        #region Filter out comments
        //CMNT
        if(cChar == '/' && cTemp == '/') {
            if(txt.Length - 1 == txtIndex)
                return result;
            while(txt[++txtIndex] != '\n') {
                if(txt.Length == txtIndex)
                    return result;
            }
            result += '\r';
            if(txt.Length - 1 != txtIndex)
                cTemp = txt[++txtIndex];
            continue;
        }
        if(cChar == '/' && cTemp == '*') {
            if(txt.Length - 1 == txtIndex)
                return result;
            cTemp = txt[++txtIndex];
            do {
                cChar = cTemp;
                if(txt.Length - 1 == txtIndex)
                    return result + cChar;
                cTemp = txt[++txtIndex];
            } while(cChar != '*' && cTemp != '/');
            result += '\r';
            if(txt.Length - 1 != txtIndex)
                cTemp = txt[++txtIndex];
            continue;
        }
        #endregion
        iState = getNextState(iState,cChar);
        switch(iState) {
            case 0:
                result += cChar;
                iState = 0;
                sToken = "";
                break;
            case 1:
            case 3:
            case 5:
            case 7:
            case 10:
            case 14:
            case 18:
            case 25:
            case 26:
                sToken += cChar;
                break;
            case 2:
```

```
                    if(isKeyword(sToken))
                        result += sToken;
                    else
                        result += "<ID>";
                    iState = 0;
                    flag=false;
                    sToken = "";
                    break;
                case 4:
                    result += "<INT>";
                    iState = 0;
                    flag=false;
                    sToken = "";
                    break;
                case 6:
                    result += "<FLOAT>";
                    iState = 0;
                    flag=false;
                    sToken = "";
                    break;
                case 8:
                    result += "<STR>";
                    iState = 0;
                    sToken = "";
                    break;
                case 9:
                case 11:
                case 12:
                case 13:
                case 15:
                case 16:
                case 17:
                case 19:
                case 20:
                case 21:
                case 22:
                case 23:
                case 24:
                case 27:
                case 28:
                    result += "<OPR>";
                    if(cChar != '+' && cChar != '-' && cChar != '/'
                        && cChar != '>' && cChar != '<' && cChar != '=')
                        flag=false;
                    iState = 0;
                    sToken = "";
                    break;
                case 30:
                case 33:
                    iState = 0;
                    sToken = "";
                    break;
            }
        }
```

```
            return result;
        }
    }

}
```

Output: