Session: 2021 – 2025

Submitted by:

Abdul Mateen 2021-CS-190

Supervised to:

Sir Laeeq Niazi

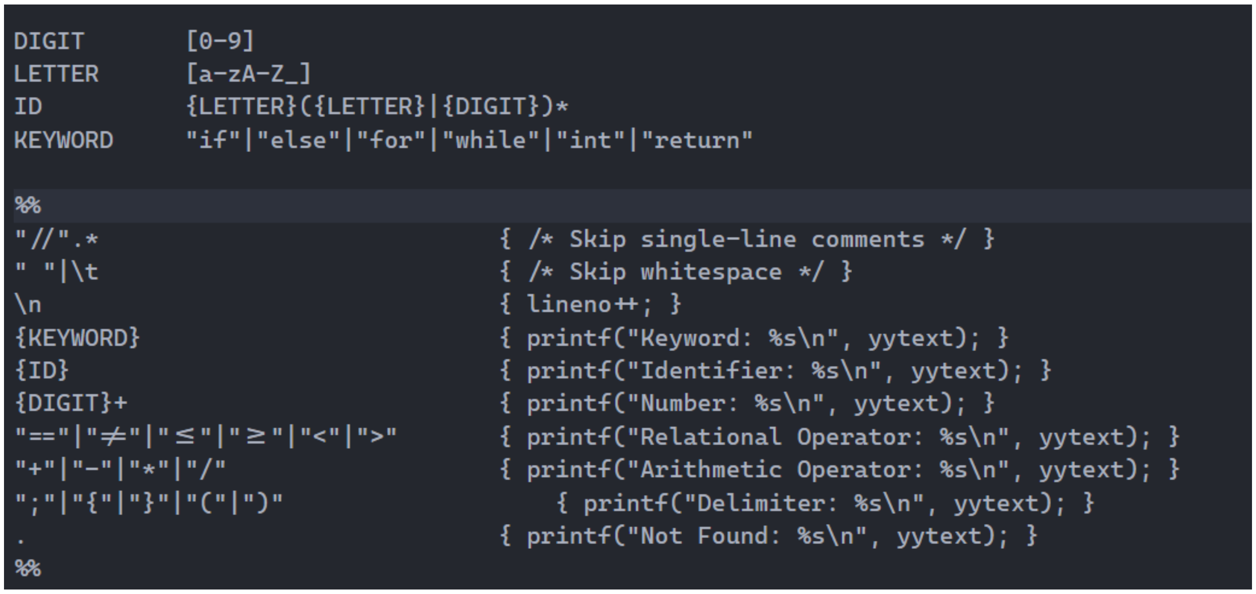
Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

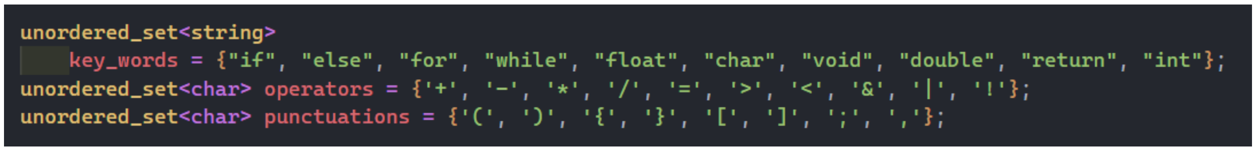
Task 1:

Define more concrete rules in the file and generate the tokens for your any recursion C++ program.

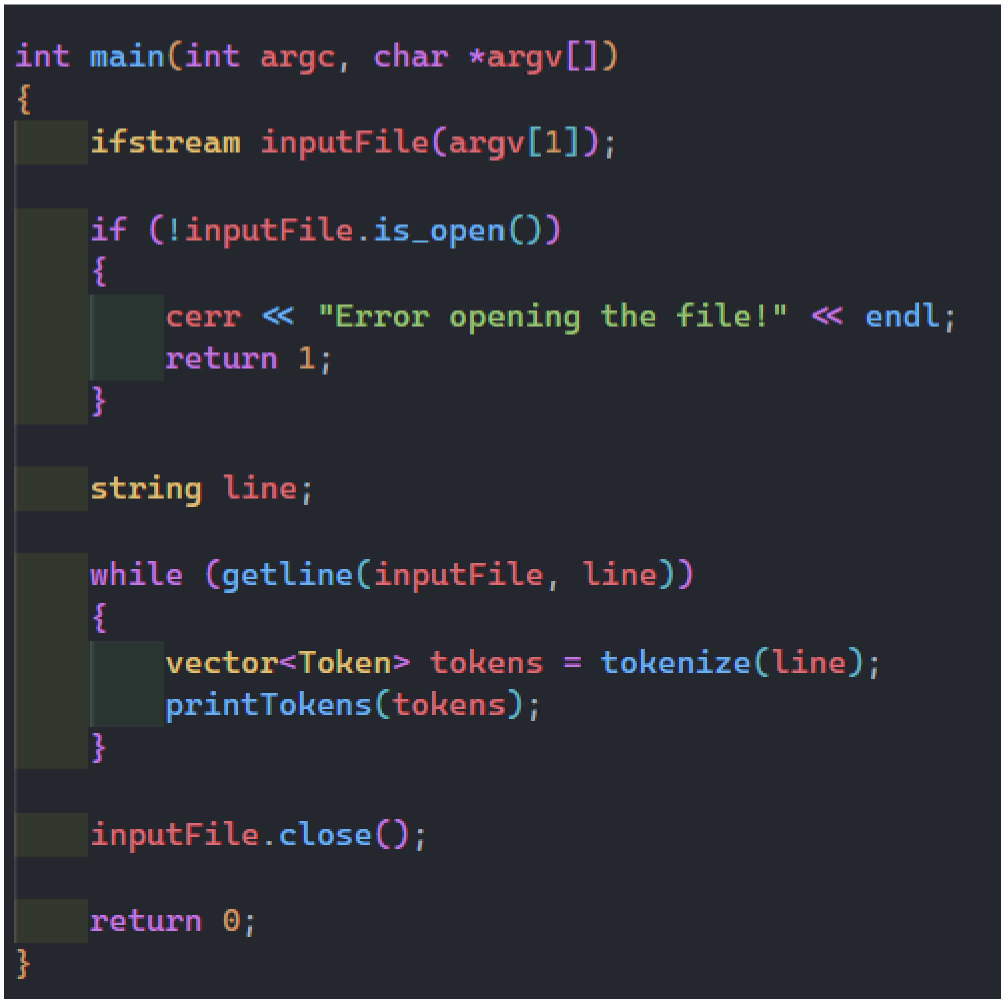


```
DIGIT       [0-9]
LETTER      [a-zA-Z_]
ID          {LETTER}({LETTER}|{DIGIT})*
KEYWORD     "if"|"else"|"for"|"while"|"int"|"return"

%%
"//".*                              { /* Skip single-line comments */ }
" "|\t                              { /* Skip whitespace */ }
\n                                  { lineno++; }
{KEYWORD}                           { printf("Keyword: %s\n", yytext); }
{ID}                                { printf("Identifier: %s\n", yytext); }
{DIGIT}+                            { printf("Number: %s\n", yytext); }
"=="|"≠"|"≤"|"≥"|"<"|">"            { printf("Relational Operator: %s\n", yytext); }
"+"|"-"|"*"|"/"                     { printf("Arithmetic Operator: %s\n", yytext); }
";"|"{"|"}"|"("|")"                     { printf("Delimiter: %s\n", yytext); }
.                                   { printf("Not Found: %s\n", yytext); }
%%
```

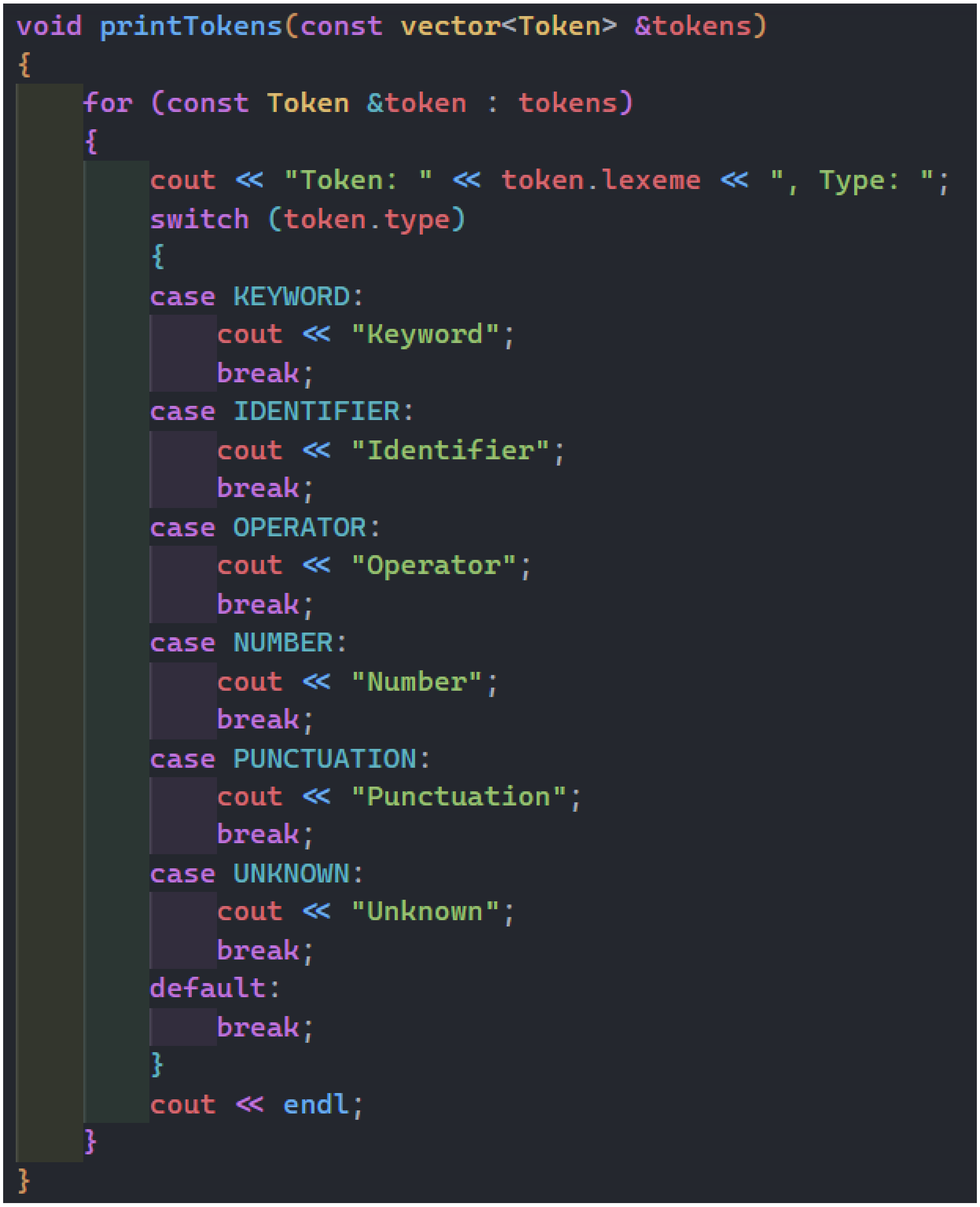
Tokenizer:



```
unordered_set<string>
    key_words = {"if", "else", "for", "while", "float", "char", "void", "double", "return", "int"};
unordered_set<char> operators = {'+', '-', '*', '/', '=', '>', '<', '&', '|', '!'};
unordered_set<char> punctuations = {'(', ')', '{', '}', '[', ']', ';', ','};
```



```cpp
int main(int argc, char *argv[])
{
    ifstream inputFile(argv[1]);

    if (!inputFile.is_open())
    {
        cerr << "Error opening the file!" << endl;
        return 1;
    }

    string line;

    while (getline(inputFile, line))
    {
        vector<Token> tokens = tokenize(line);
        printTokens(tokens);
    }

    inputFile.close();

    return 0;
}
```



```cpp
void printTokens(const vector<Token> &tokens)
{
    for (const Token &token : tokens)
    {
        cout << "Token: " << token.lexeme << ", Type: ";
        switch (token.type)
        {
        case KEYWORD:
            cout << "Keyword";
            break;
        case IDENTIFIER:
            cout << "Identifier";
            break;
        case OPERATOR:
            cout << "Operator";
            break;
        case NUMBER:
            cout << "Number";
            break;
        case PUNCTUATION:
            cout << "Punctuation";
            break;
        case UNKNOWN:
            cout << "Unknown";
            break;
        default:
            break;
        }
        cout << endl;
    }
}
```

Time Complexity:

The time complexity of tokenizer is O(n).