Kingdom of Saudi Arabia
Ministry of Education
Shaqra University
College of Computing & Information Technology

المملكة العربية السعودية
وزارة التعليم
جامعة شقراء
كلية الحاسب الالي و تقنية المعلومات

LAB – I
Compiler Design

Lexical Analysis (Token Generation)

MAR 2022 – JUNE 2023

DEPARTMENT OF COMPUTER SCIENCE

COLLEGE OF COMPUTING AND INFORMATION TECHNOLOGY

SHAQRA UNIVERSITY

P.O. BOX 15572

Dr. Nayyar Ahmed Khan

EMAIL: nayyar@su.edu.sa

Lexical Analysis

The first phase of scanner works as a text scanner. This phase scans the source code as a stream of characters and converts it into meaningful lexemes. Lexical analyzer represents these lexemes in the form of tokens as:

<token-name, attribute-value>

Syntax Analysis

The next phase is called the syntax analysis or parsing. It takes the token produced by lexical analysis as input and generates a parse tree (or syntax tree). In this phase, token arrangements are checked against the source code grammar, i.e. the parser checks if the expression made by the tokens is syntactically correct.

Code to demonstrate the breaking of User Text File into various segments such as Keywords / Operators / Delimiters

```
Lab Exercise I : Demonstrative Code for Compiler Design Lab I

keywords = {"auto","break","case","char","const","continue","default","do",

"double","else","enum","extern","float","for","goto",

"if","int","long","register","return","short","signed",

"sizeof","static","struct","switch","typedef","union",

"unsigned","void","volatile","while","printf","scanf","%d","include","stdio.h

","main"}


operators = {"+","-","*","/","<",">","=","<=",">=","==","!=","++","--","%"}


delimiters = {'(',')','{','}','[',']','"',"'",';','#',',',''}
```

```
def detect_keywords(text):
        arr = []
        for word in text:
                if word in keywords:
                        arr.append(word)
        return list(set(arr))

def detect_operators(text):
        arr = []
        for word in text:
                if word in operators:
                        arr.append(word)
        return list(set(arr))

def detect_delimiters(text):
        arr = []
        for word in text:
                if word in delimiters:
                        arr.append(word)
        return list(set(arr))

def detect_num(text):
        arr = []
        for word in text:
                try:
                        a = int(word)
```

```
            arr.append(word)
        except:
            pass
    return list(set(arr))

def detect_identifiers(text):
    k = detect_keywords(text)
    o = detect_operators(text)
    d = detect_delimiters(text)
    n = detect_num(text)
    not_ident = k + o + d + n
    arr = []
    for word in text:
        if word not in not_ident:
            arr.append(word)
    return arr

with open('D:/Nayyar/Compiler Design/myText.txt') as t:
    text = t.read().split()

print("Keywords: ",detect_keywords(text))
print("Operators: ",detect_operators(text))
print("Delimiters: ",detect_delimiters(text))
print("Identifiers: ",detect_identifiers(text))
print("Numbers: ",detect_num(text))
```

General Questions to answer:

1. What is Lexical Analysis?

…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………

2. Why the Compiler need to break up the user input in various tokens?

…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………

3. Modify the code above to check special characters in your program:
[√ , θ , £ , ¥ , © , ® , ± , ≠ , ≤ , ≥ , ÷ , ∞ , μ , α , β]?

…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………
…………………………………………………………………………………………