Part – A

1)C
2)B
3)B
4)C
5)D

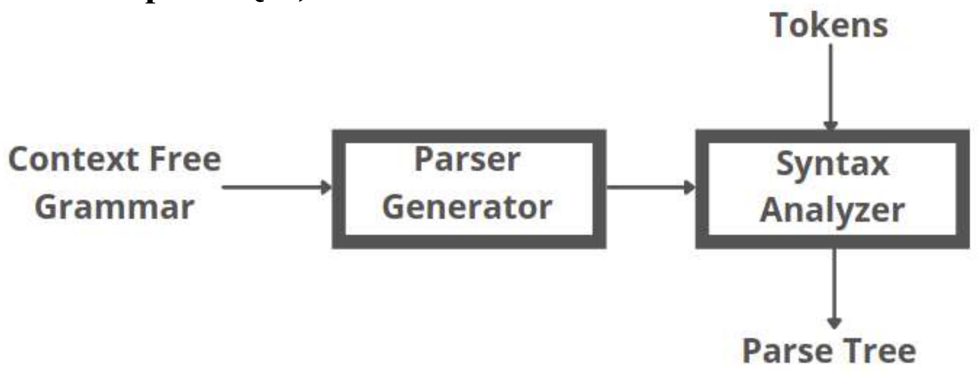
Part – B

6) State the complier construction tools. Explain them

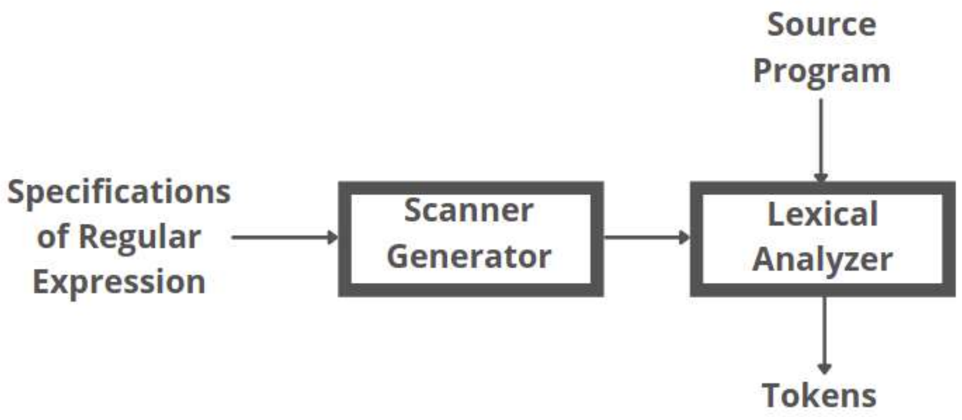
A compiler is a computer program that converts source code written in a computer language (the source language) into another computer language (the target language, providing having a binary form referred to as object code). The best reason for inadequate to convert source code is to create an executable code.

Parser Generator

- **Parser Generator** produces syntax analyzers (parsers) based on context-free grammar that takes input in the form of the syntax of a programming language. It's helpful because the syntax analysis phase is quite complex and takes more compilation and manual time.
- **Example:** EQM, PIC



- **Scanner Generator**
- Scanner Generator generates lexical analyzers from the input that consists of regular expression descriptions based on tokens of a language. It generates a finite automaton to identify the regular expression.
- **Example:** LEX is a scanner generator provided by UNIX systems.



- **Syntax Directed Translation Engines**

  Syntax Directed Translation Engines take a parse tree as input and generate intermediate code with three address formats. These engines contain routines to traverse the parse tree and generate intermediate code. Each parse tree node has one or more translations associated with it.

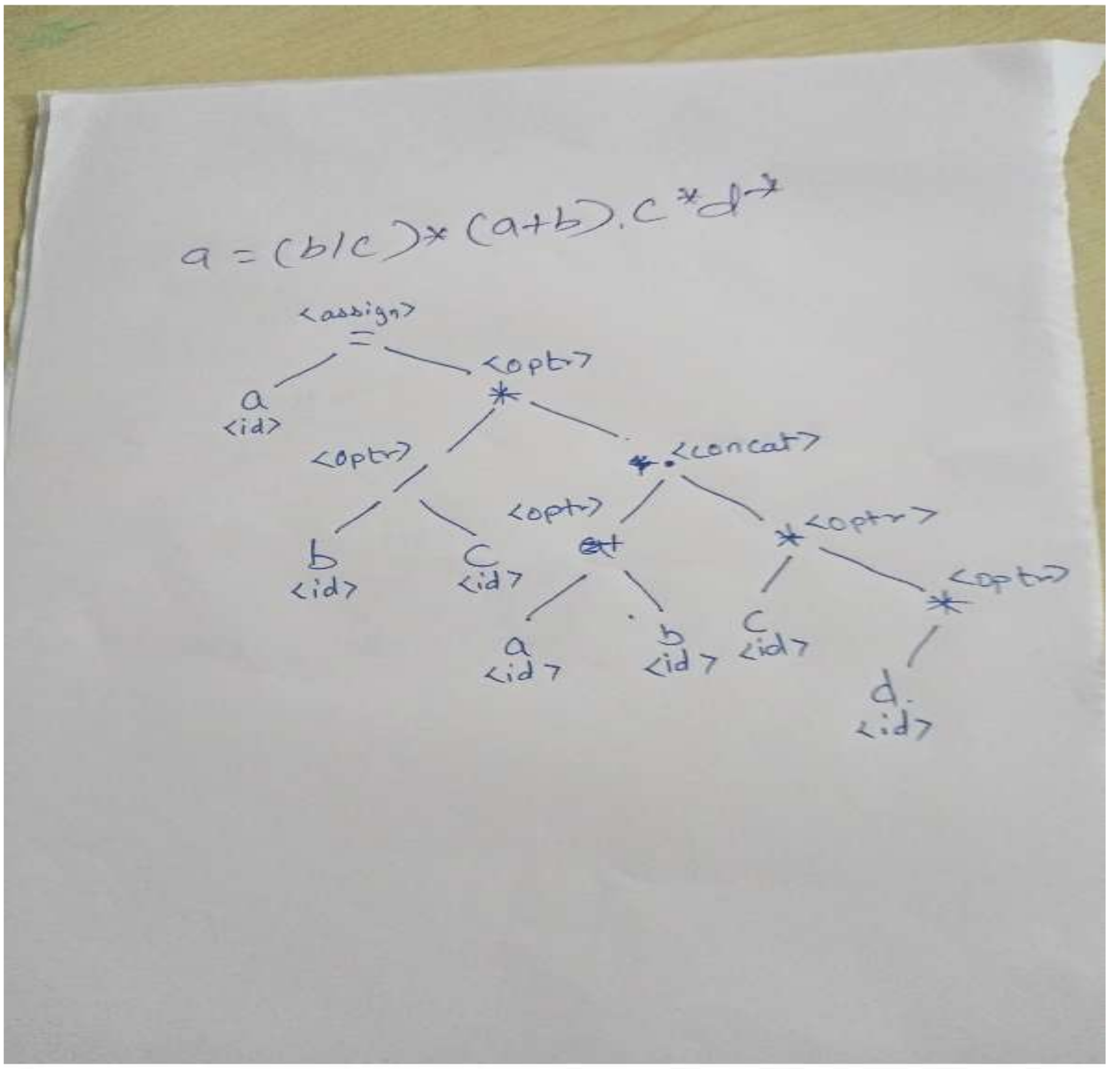
- **Automatic Code Generators**

  Automatic Code Generators take intermediate code as input and convert it into machine language. Each intermediate language operation is translated using a set of rules and then sent into the code generator as an input. A template matching process is used, and by using the templates, an intermediate language statement is replaced by its machine language equivalent.

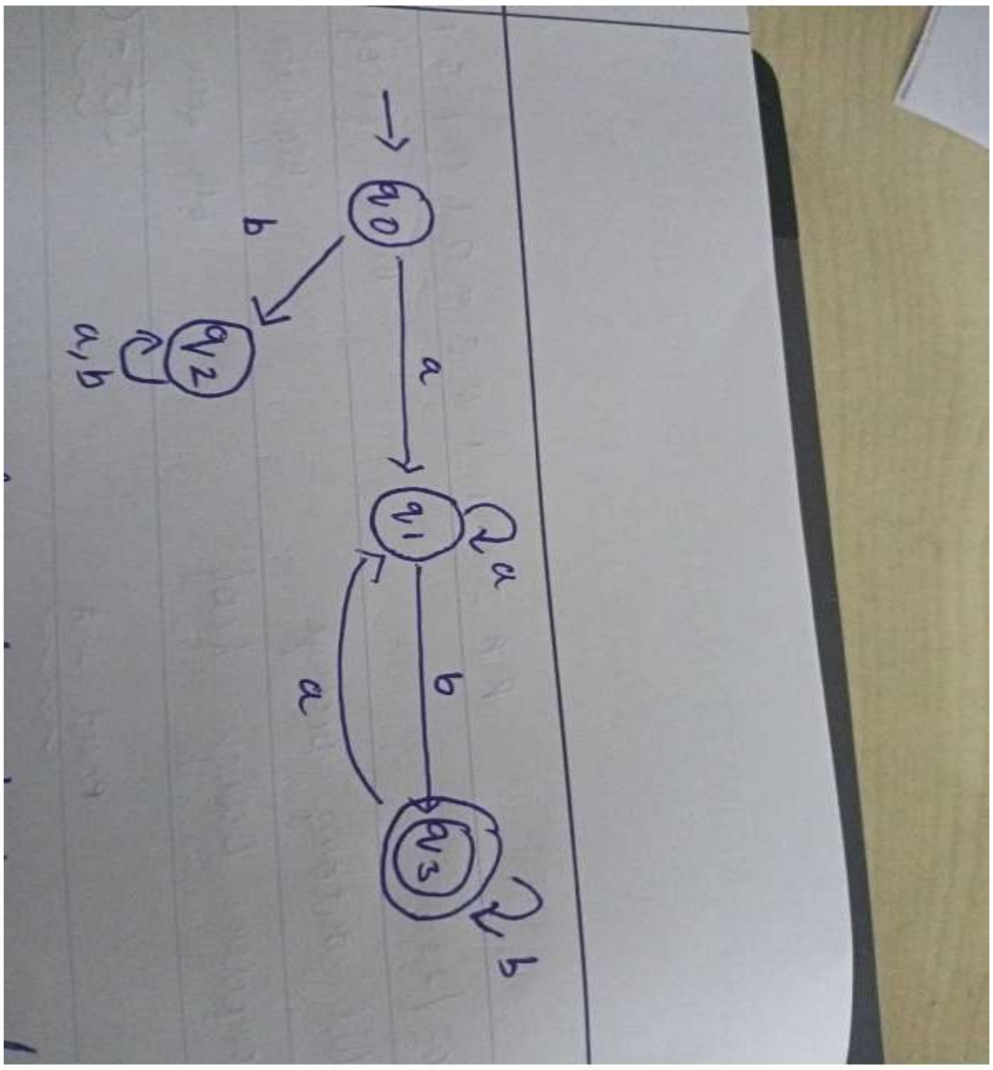
- **Data-Flow Analysis Engines**

  Data-Flow Analysis Engines is used for code optimization and can generate an optimized code. Data flow analysis is an essential part of code optimization that collects the information, the values that flow from one part of a program to another.

7) What is syntax tree?.Draw the syntax tree for the expression a=(b/c)*(a+b)c*d*



8) Design a Deterministic Finite Automata (DFA) to accept strings that begin with a and end with b over Σ={a,b}. Write the formal definition of the DFA



Part – C

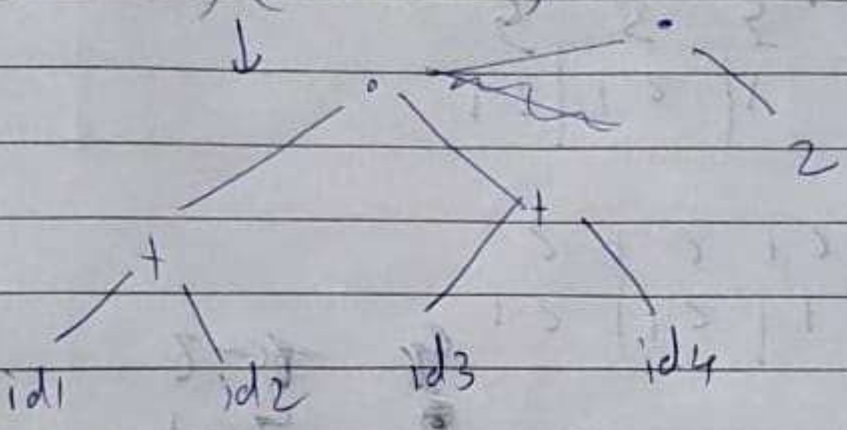
9) Explain the phases of compiler and draw the translation of statement (b+c)(b+c)2

$(b+c)(b+c)\cdot 2$

$\downarrow$

$(id1 + id2)(id3 + id4) \cdot 2$

$\downarrow$



$\downarrow$

$t_1 = id1 + id2$

$t_2 = id3 + id4$

$t3 = t1 \cdot t2$

$t4 =$ [illegible] 2

$t5 = t3 \cdot t4$

$\downarrow$

$t1 = id1 + id2$

$t2 = id3 + id4$

$t3 = t1 \cdot t2$

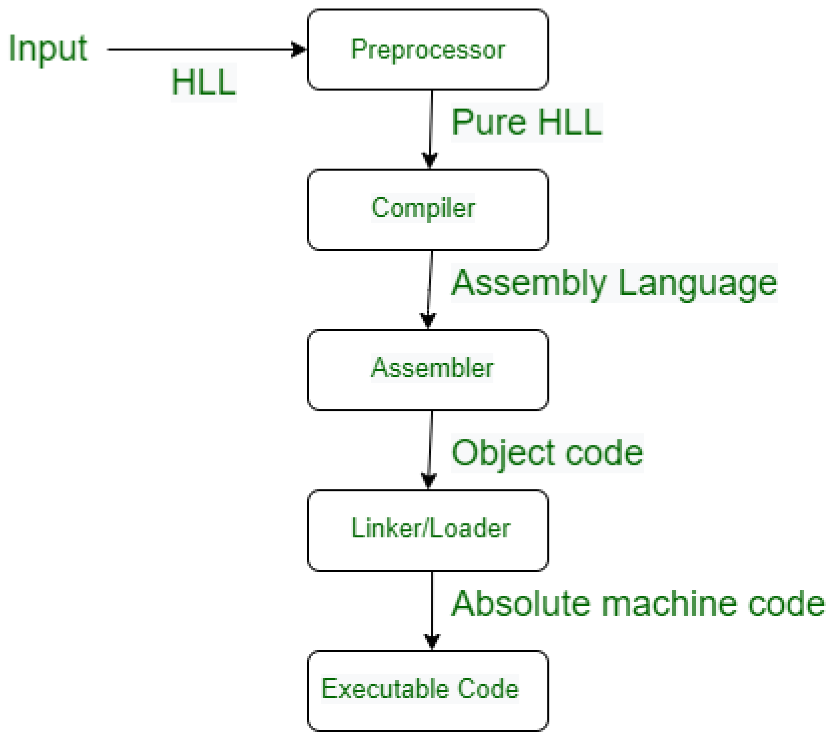
$t4 = t3 \cdot 2$

$\downarrow$

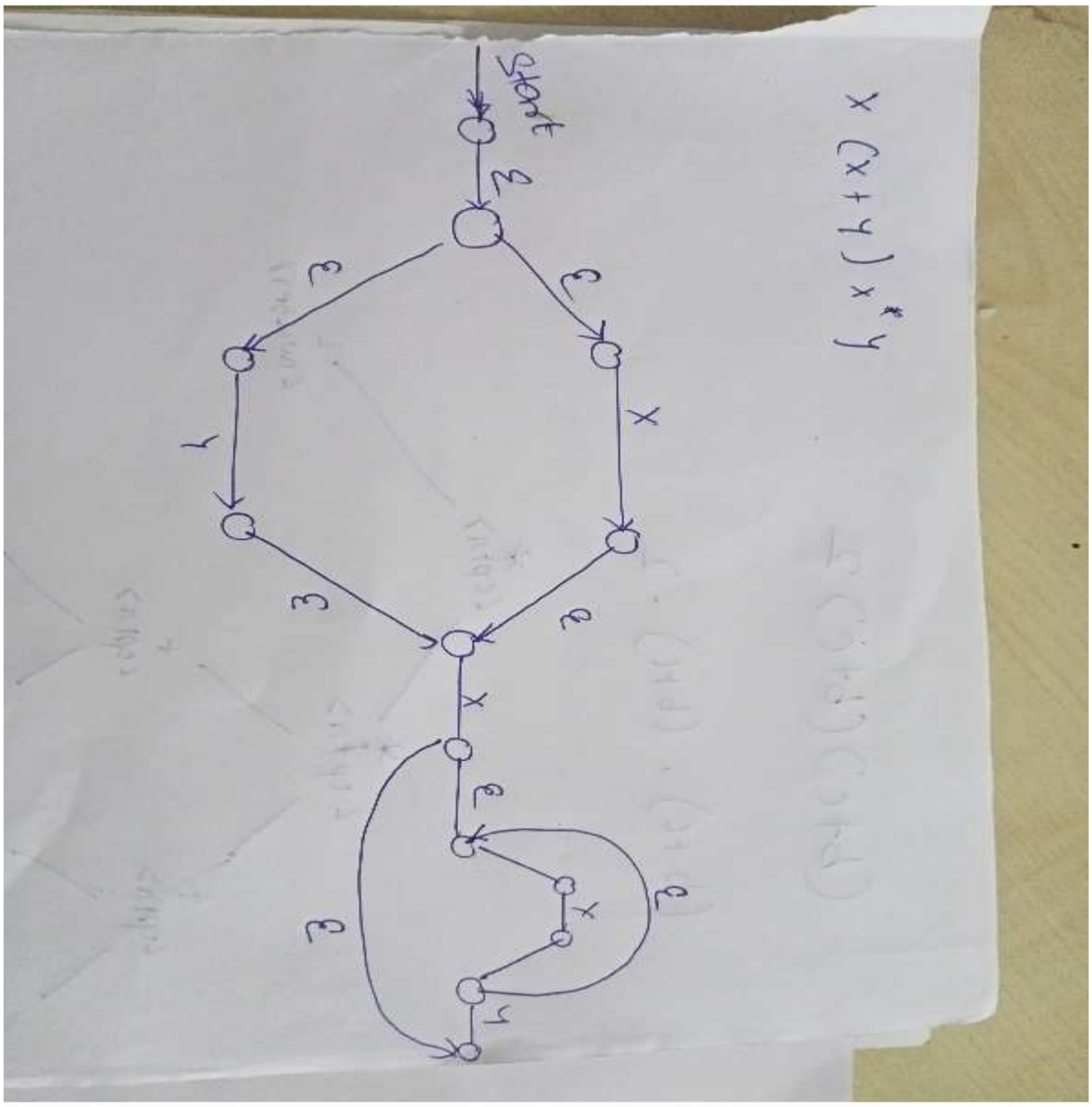
```
LOAD  R1 , id1
ADD R3, R1 , id2
LOAD  R4 , id3
[illegible]  id4
```

b) Draw the language processing system with neat diagram

.



10)Interpret a DFA for the given RE= x(x+y)x*y **using Direct Method and Discuss the input buffering techniques in detail**



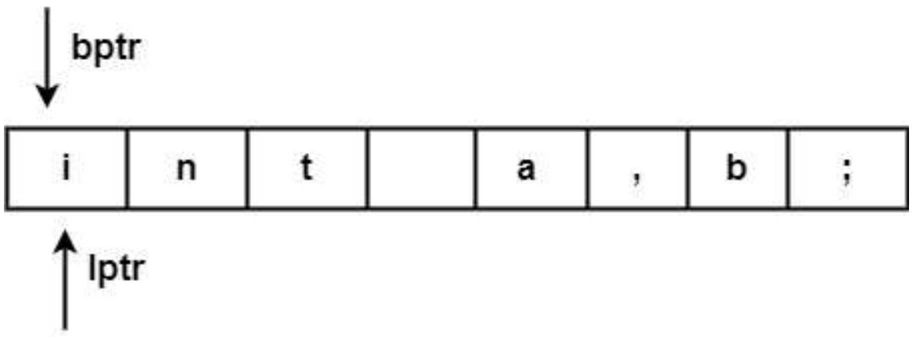
Lexical Analysis has to access secondary memory each time to identify tokens. It is time-consuming and costly. So, the input strings are stored into a buffer and then scanned by Lexical Analysis.

Lexical Analysis scans input string from left to right one character at a time to identify tokens. It uses two pointers to scan tokens −

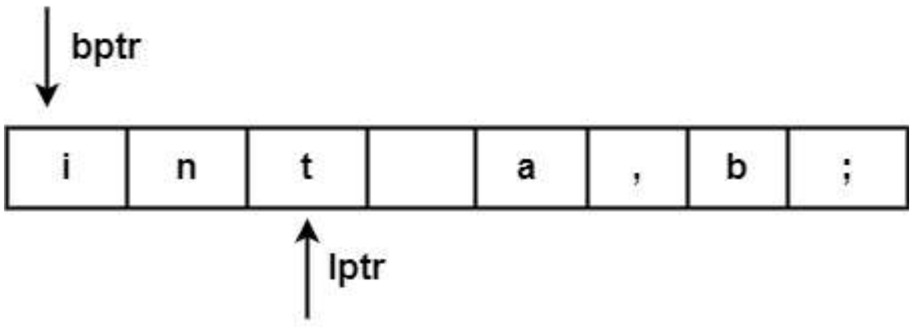
- **Begin Pointer (bptr)** − It points to the beginning of the string to be read.
- **Look Ahead Pointer (lptr)** − It moves ahead to search for the end of the token.

Example − For statement int a, b;

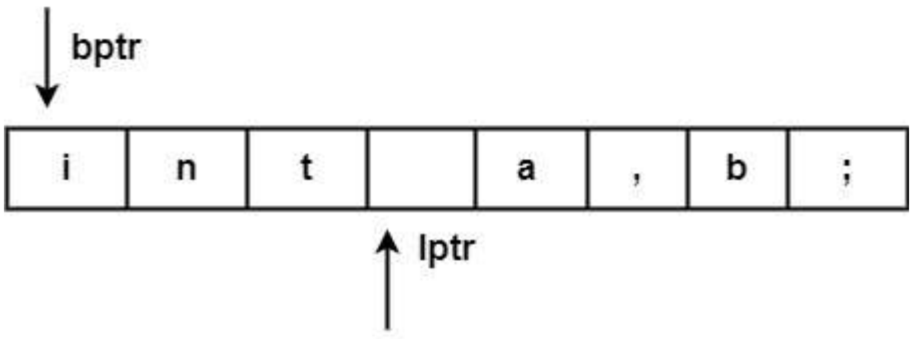
- Both pointers start at the beginning of the string, which is stored in the buffer.



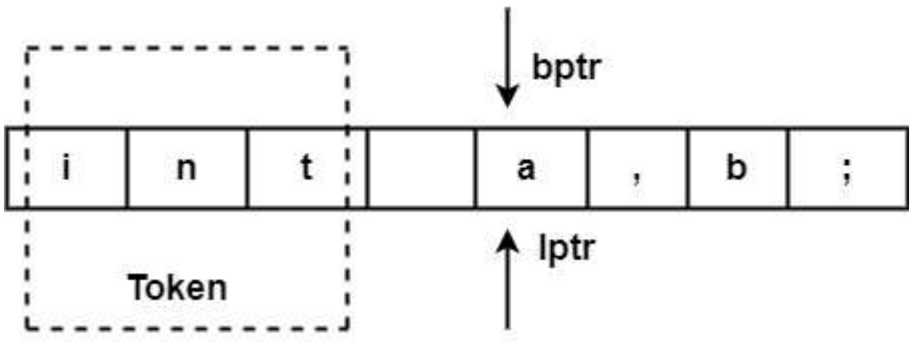
- Look Ahead Pointer scans buffer until the token is found.



- The character ("blank space") beyond the token ("int") have to be examined before the token ("int") will be determined.



- After processing token ("int") both pointers will set to the next token ('a'), & this process will be repeated for the whole program.



A buffer can be divided into two halves. If the look Ahead pointer moves towards halfway in First Half, the second half is filled with new characters to be read. If the look Ahead pointer moves towards the right end of the buffer of the second half, the first half will be filled with new characters, and it goes on.

Advantages

- It usually just does one test to determine if the forward pointer is pointing to an eof.

- It only runs further tests until it reaches the halfway point of the buffer or eof.
- The average number of tests per input character is extremely close to 1 since N input characters are encountered between eofs.