Compiler Design

Compiler : Execute whole code in one time
↳ C, C++, ... → Language for software development

Interpreter : Line by line execution
↳ Small Code Use → Web based Language

→ Compiler execute whole program at a time
↳ Large Code Use

ex: C, C++, Java, Pascal, COBOL, BASIC
↳ Programming language

* Interpreter example : Ruby, Python, ASP.NET, Perl
↓
Scripting language

COBOL → Common Business oriented Language

BASIC → Beginner All Purpose symbolic Instruction Code.

* Markup Language → HTML, XML → No use they just help in ~~Interpreter~~ with scripting language

* Grammar : 4 Tuple

(V , T , P , S)

- V → Non Terminal
- T → Terminal
- P → Productional rule
- S → Starting Symbol

① Terminal :- Small Letter
(a-z) (0-9)

② Non Terminal :-
(A-Z)

③ Production Rule :- S → abc
↓
P.R.

④ Starting Symbol :- S → abc
↓
S.S.

Regular Grammar (R.G.) → Finite length.
- Least Powerful

↳ 4 Tuple (V, T, P, S)

↳ Two form → Left linear Regular Grammar **
→ Right linear Regular Grammar **

① Left linear Regular Grammar :-

→ Non Terminal on left

S → Aaa

S → Bba

S → Xaaa

(2) Right Linear Regular Grammar :-

$S \longrightarrow aa\,A\boxed{A}$
$S \longrightarrow aB\,a\boxed{B}$
$S \longrightarrow aX\,aa\,X\boxed{B}$

★ $S \longrightarrow \boxed{A}aa\boxed{A}$
$S \longrightarrow \boxed{B}\boxed{B}$
$S \longrightarrow \boxed{Z}X\boxed{A}\boxed{B}$
$S \longrightarrow \boxed{A}aa\,X\,a\boxed{B}$
→ Left, Right both

Q. N.T.
$S \longrightarrow aa$ ✗
$S \longrightarrow \epsilon$ ✗
$S \longrightarrow aaaa$ ✗
→ R.G. ✗

★ If it is R.G. then it will be C.F.G., C.S.G., Unrestricted Grammar

Q. (i) $S \longrightarrow aA\boxed{A}$
$S \longrightarrow aab$
$A \longrightarrow a$
→ Right linear grammar
R.G. ✓

(ii) $S \longrightarrow aaAa$
$S \longrightarrow \boxed{A}\boxed{A}$
$S \longrightarrow B$
$S \longrightarrow a$
$B \longrightarrow b$
Both linear grammar
Regular Grammar

(iv) $S \rightarrow aAa$
$S \rightarrow aBa$
R.G. ×

Q. (i) $A \rightarrow aaAaA$
$B \rightarrow bb$
} R.G. ✓ (Right Linear)

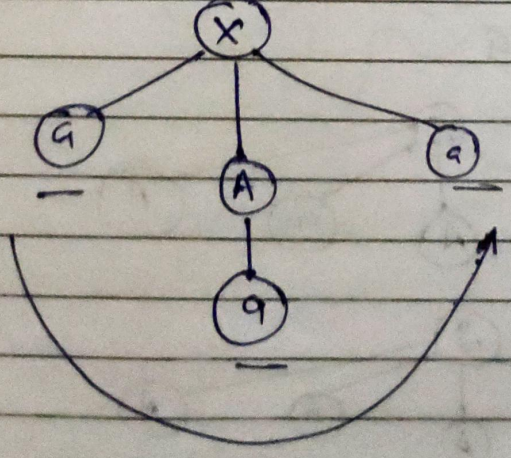
(ii) $B \rightarrow b$
$A \rightarrow aAa$
$A \rightarrow aa$
} R.G. ×

(iii) $A \rightarrow aAA$
$B \rightarrow BB\epsilon$ (ε = Empty)
} Both Linear R.G. ✓

(1) Derivation Tree :-

Starting Node → Root Node
Vertex → Non Terminal
Leaf Node (leaves) → Terminal

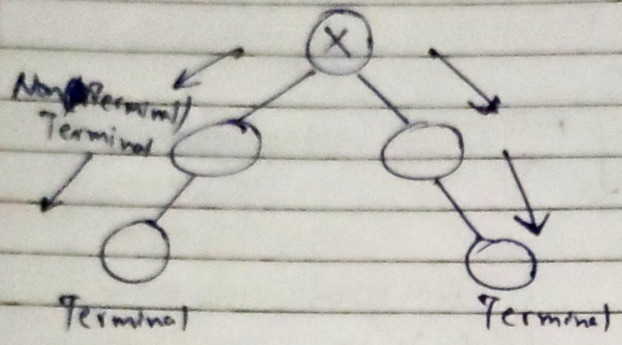
$X \rightarrow aAa$
$A \rightarrow a$



⇒ aaa

(i) Top-down derivation tree -

(i) Start with Root Node.
(ii) Goes down Toward leaf Node (leaves)



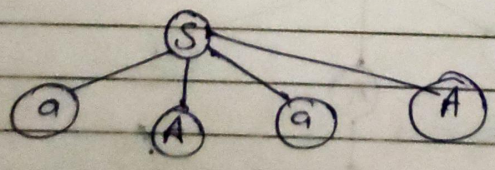
(ii) Bottom-up derivation tree :-

(i) Start with leaf Node (leaves Node)
(ii) Towards Root Node.

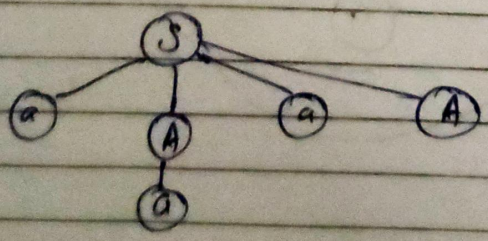
(i) Left linear derivation tree :-

Root Node ← S → aAaA
A → a

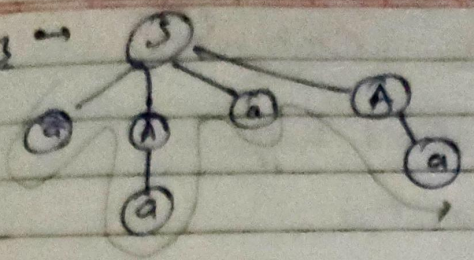
Step 1 →



Step 2 →



Step 3 :-



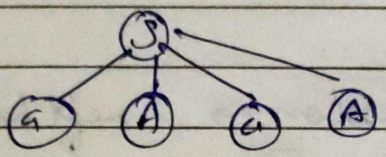
$S \rightarrow aaaa$

(ii) Right linear derivation tree :- (Right to left)

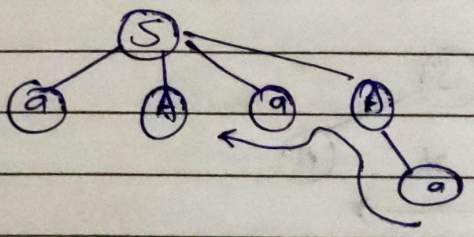
$S \rightarrow aA\,aA$

~~$A \rightarrow a$~~

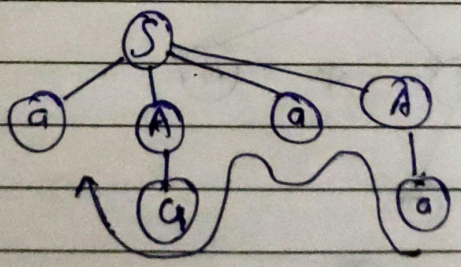
Step 1 :-



Step 2 :-



Step 3 :-



$S \rightarrow aaaa$

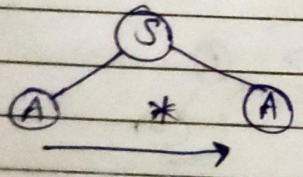
Ambiguity Grammar (Ambiguous)

(Any one should happen)

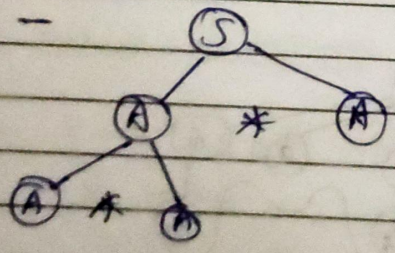
(i) Left linear derivation tree
(ii) Right " " "
(iii) Left and Right " " "

Q. (i) $S \longrightarrow A * A$
(ii) $A \longrightarrow A + A$
(iii) $A \longrightarrow A * A$
(iv) $A \longrightarrow a$
(v) $B \longrightarrow b$

Take any expression → left linear →
we take (i)

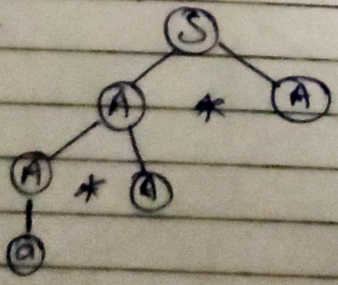


Step 1 –



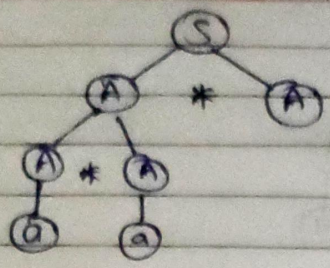
we use (iii) now

Step 2 :-

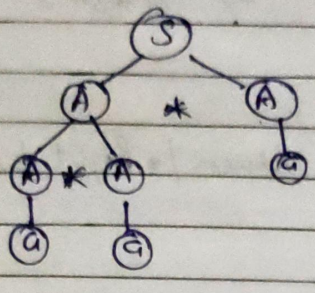


we use (iv) now

Step 3:-

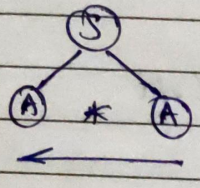


Step 4:-

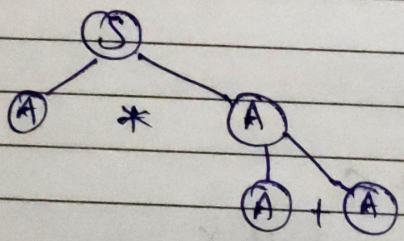


⇒ a * a * a

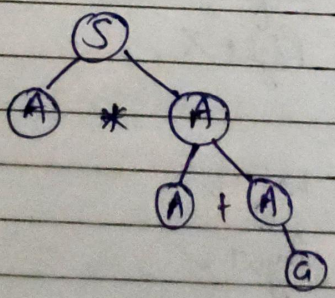
Right Linear :-



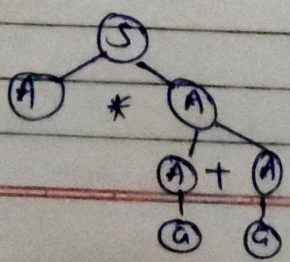
Step 1 :-



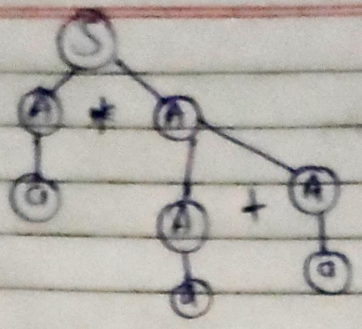
Step 2 :-



Step 3 :-



Step 4:



→ a*a+a

Q Find out left linear / Right linear Regular Gram.

① S → aaAB
S → aABB Right
A → a
B → b

② S → AAaa Left
S → BBb
A → a
B → b

③ S → ∈ + ∈

Left ✗
Right ✗

④ S → AaaA
S → BA
A → a
B → b

Left ✓
Right ✓

Power of Sigma :-

$\Sigma = (a, b)$

(i) $\Sigma^0 \longrightarrow$ only Empty string
$\Sigma^0 \longrightarrow$ Empty

(ii) $\Sigma^1 \longrightarrow$ Set of all string of length 1
$\Sigma^1 \longrightarrow (a, b)$

(iii) $\Sigma^2 \longrightarrow$ Set of all string of length 2
$\Sigma^2 \longrightarrow (aa, ab, ba, bb)$

(iv) $\Sigma^3 \longrightarrow$ Set of all string of length 3
$\Sigma^3 \longrightarrow (aaa, bbb, abb, baa, bab, aba, aab, bba)$

(v) $\Sigma^4 \longrightarrow$ Set of all string of length 4
$\Sigma^4 \longrightarrow (aaaa, \ldots)$

(vi) $\Sigma^n \longrightarrow$ Set of all string of length n

① Kleene Closure (*)

$\Sigma = (a, b)$

$\Sigma^* = (\text{Empty}, a, b, aa, ba, \ldots)$

② Kleene Plus (+)

$\Sigma = (a, b)$

$\boxed{\Sigma^+ = \Sigma^* - (\text{Empty})}$

$\Sigma^+ = (a, b, aa, bb, \ldots)$

Regular Language :- ('L')

↳ Language accepted by Regular grammar is called Regular Language.

$\Sigma = (a, b)$

$a^* = (\text{Empty}, a, aa, aaa, \ldots)$

$a^+ = (a, aa, aaa, \ldots)$

$\Sigma = (a, b)$

$L^1 = (a, b)$
$L^2 = (aa, bb, ab, ba)$
$L^3 = (aaa, aab, abb, bbb, \ldots)$
$L^4 = (aaaa, aaab, aabb, abbb, \ldots)$

$\Sigma = (a, b, c, d)$

$L^1 = (a, b, c, d)$
$L^2 = (ab, ac, ad, ba, bd, \ldots)$
$L^3 = (abc, bcd, \ldots)$

* **Five Properties of Regular Language :-**

① **Union** – if we have 2 Regular language L_1 and L_2. Then Union of $L_1 \cup L_2$ are Regular Language.

Ex. $L_1 = (a, b)$
$L_2 = (a, c)$
$L_1 \cup L_2 = (a, b, c)$

② **Complement** – if we have 2 Regular language L_1 and L_2. Then Complement of L_1' and L_2' are Regular Language.

$L_1 = (a, b)$ $\quad L_1' = (b, a)$
$L_2 = (a, c)$ $\quad L_2' = (c, a)$

③ **Concatenation :-** if we have 2 regular language L_1 and L_2. Then concatenation of $L_1 \cdot L_2$ are regular language.

$L_1 = (a, b)$
$L_2 = (a, c)$

$L_1 \cdot L_2 = (aa, ac, ba, bc)$

④ **Kleene Closure (*)** :- if we have 2 Regular language L_1 and L_2. Then Kleene closure of L_1^* and L_2^* are Regular Language

$L_1 = (a, b)$
$L_2 = (a, c)$

$L_1^* = (\text{Empty}, a, b, aa, bb, ab, ba, \dots)$
$L_2^* = (\text{Empty}, a, c, aa, ac, cc, ca, \dots)$

⑤ **Intersection ($\cap$)** :- if we have 2 Regular language L_1 and L_2. Then intersection of $L_1 \cap L_2$ are regular language.

$L_1 = (a, b)$
$L_2 = (a, c)$

$L_1 \cap L_2 = (a)$

★ **Precedence** →

① Kleene Closure (*) Highest Priority

② ~~Second~~ Concatenation

③ Union (lowest)

PAGE NO.
DATE / /

Normal Form

CNF (Chomsky Normal Form)

GNF (Greibach Normal form)

1) CNF :-

Conditions -

(i) Non-Terminal Represent to empty symbol.

$A \longrightarrow \epsilon$

(ii) Non-Terminal Represent to Terminal.

$A \longrightarrow b$, $A \longrightarrow c$

(iii) Non-Terminal represent atleast to 2 Non Terminal
$(V \rightarrow V_1 V_2)$

$A \rightarrow BC$, $A \rightarrow AB$, $A \rightarrow BD$

2) GNF :-

Conditions -

(i) Non-Terminal Represent to empty symbol.

$A \longrightarrow \epsilon$

(ii) Non-Terminal Represent to Terminal

$A \rightarrow b$, $A \longrightarrow c$

(iii) Non-Terminal Represent Terminal followed by Non Terminals. (aA^*)

$A \to aB$, $A \to aBC$, $A \to aBCD$, $A \to aABCD$

↓

Only one terminal followed by terminals

(i) $A \to d$ ✓

$A \to \epsilon$ ✓ CNF ✗

$A \to Aa$ ✗ GNF ✗

$A \to a$ ✓ Regular Grammar ✓ (VTPS)

(ii) $A \to a$ ✓

$A \to \epsilon$ ✓ CNF ✓

$A \to BC$ ✓ GNF ✗

(iii) $A \to a$ ✓

$A \to b$ ✓ CNF ✗

$A \to aA$ ✗ GNF ✗

$A \to 6$ ✓

(iv) $A \to a$ ✓

$A \to \epsilon$ ✓ CNF ✗

$A \to aBCDEFGHIJ$ ✓ GNF ✓

(v) $A \to a$ ✓

$A \to \epsilon$ ✓ CNF ✓

$A \to BC$ ✓ GNF ✗

(vi) $A \to \in$ ✓
$A \to b$ ✓
$A \to aBC$ ✓

CNF ×
GNF ✓

CFL (Context free Language) :-

Properties :-

(i) Union : if we have 2 Context free language L_1 and L_2 then union of ~~Context free~~ $L_1 \cup L_2$ are CFL.

(ii) Complement :- if we have 2 Context free language L_1 and L_2 then complement of L_1' and L_2' are ~~CFL~~ NOT CFL.

(iii) Concatenation :- if we have 2 CFLs L_1 and L_2. then concatenation of $L_1 \cdot L_2$ are CFL.

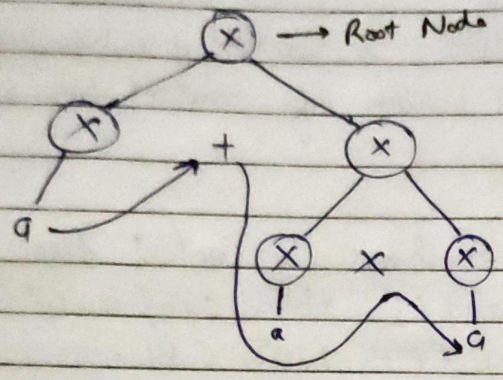
(iv) Kleene Closure (*) :- if we have 2 CFLs L_1 and L_2. then Kleene Closure L_1^* and L_2^* are CFL.

(v) Intersection :- if we have 2 CFLs L_1 and L_2 then intersection of $L_1 \cap L_2$ are NOT CFL.

(Parse Tree)

Q. Find out Left Most Derivation tree
Find string "a+a*a"

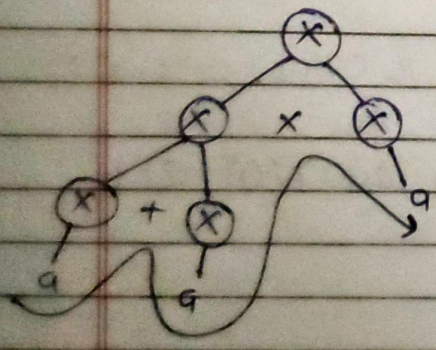
$X \rightarrow X+X$ — ①
$X \rightarrow X$
$X \rightarrow X*X$ — ③
$X \rightarrow a$



a+a*a

Right
Q. ~~Left~~ Most Derivation tree →
(Parse tree)

$X \rightarrow X+X$ — ③
$X \rightarrow X$
$X \rightarrow X*X$ ①
$X \rightarrow a$



a+a*a

★ If more than one parse tree are possible, like in this example left & Right derivation tree are

different, therefore this is Ambiguous Grammar

* If the structure of Left and Right derivation tree are same then grammar is Unambiguous grammar

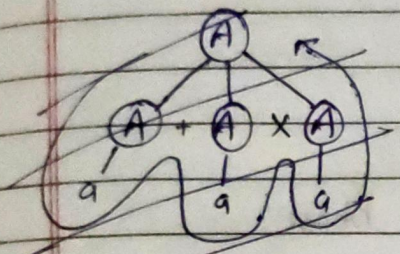
Q. Find out left most derivation tree and right.

$A \longrightarrow A+A*XA \Rightarrow A \longrightarrow A+(AXA)$

$A \longrightarrow AXA+A$

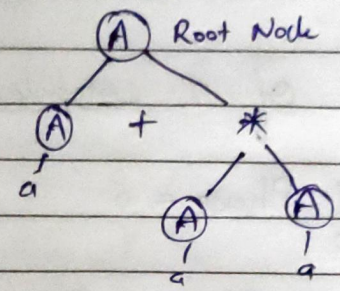
$A \longrightarrow a$

Left most →



a+axa

Sahi tha yeh bhi



diff. structure & diff. strings → Ambiguous

* For Unambiguous –

(i) ek se jyada left ban jaayen

(ii) " " " right " "

(iii) ek se jyada dono ban jaayen

Ambiguous	Unambiguous
(i) More then one left and right derivation tree.	(i) Only one left and one right derivation tree.
(ii) Different structure of Derivation tree.	(ii) Same structure of derivation tree.
(iii) (1) More than one left derivation tree. (2) More than one right derivation tree. (3) More than one left and right derivation tree.	

Phases of a compiler -

* No. of Phase = 6
* First 3 Phase called Analysis Phase.
* Last 3 Phase called Synthesis Phase.

Analysis Phase -

(i) Lexical Phase

(ii) Syntax Phase

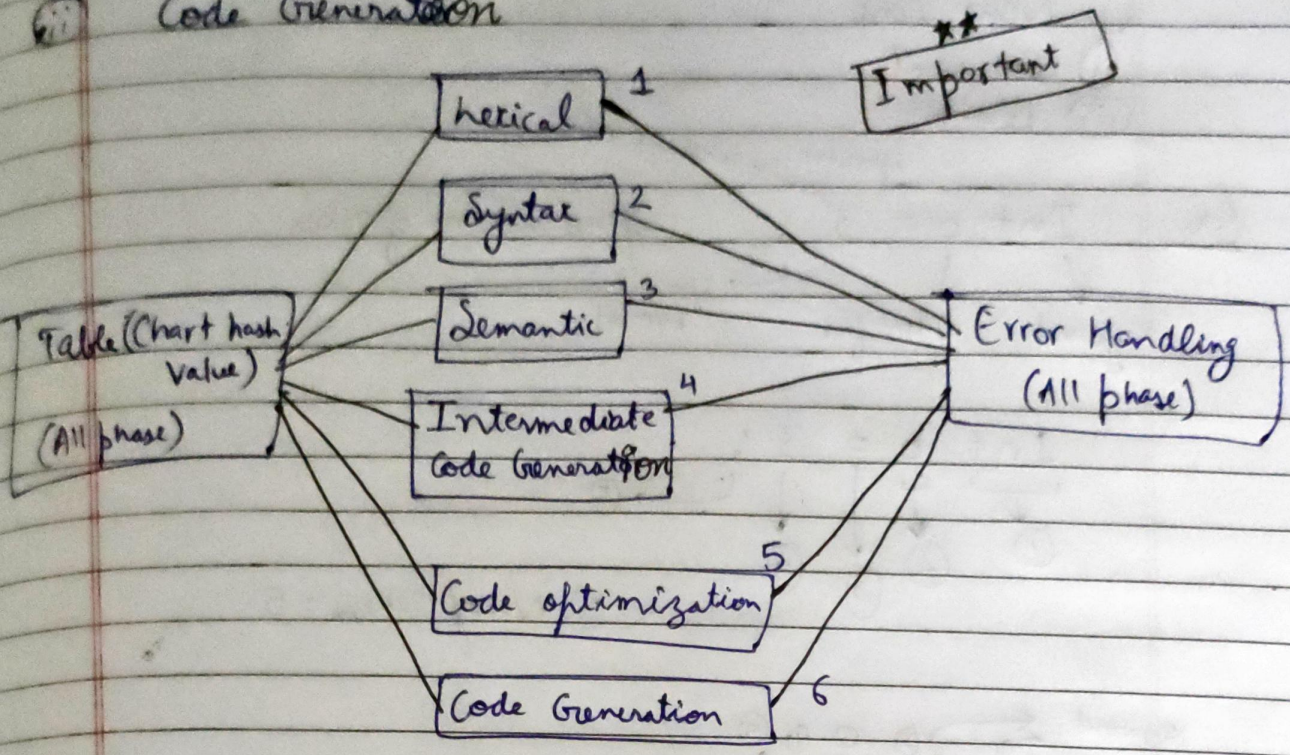
(iii) Semantic Phase

Synthesis Phase -

(i) Intermediate Code Generation

(ii) Code optimization

(iii) Code Generation



★ Token :- Sequence of character which represent unit of Information in the code.

(i) Identifier :- (a, b)

(ii) Keyword → INT, VOID

(iii) Constant value → 5, 6, 0, 1

(iv) Special Symbol → , ;

(v) Operator → Bitwise, Arithmetic

☆ Non-Token :-

(i) New line
(ii) Comment
(iii) Blank

Ex. `Int b = 3 ;` find out no. of Token

operator, constant value, Keyword, Identifier, special symbol

`Int b = 3 ;`

① ② ③ ④ ⑤

⇒ No. of Tokens = 5

Ex.

```
Int main ( ) {
Printf ( " My Name is ____ " ) ;
return 0 ;
}
```

keyword; ① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

⇒ ⑭ ☆