Compiler Design

Today's Class Topics

-Left recursion and how to remove left recursion
-Left factoring and left factored grammar
-Parser classification
-First Function
-Follow Function

Compiler Design

Remove Left Recursion

Ex:- $S \to Sa \mid Ab$

$A \to aa \mid b$

$A \to A\alpha \mid \beta$ ⇒ $A \to BA'$, $A' \to \alpha A' \mid \in$

Soln: S, A

1) $S \to Sa \mid Ab$

$S \to AbS'$

$S' \to aS' \mid \in$

2) $A \to aa \mid Sb$

⇒ non left Recursive

$S \to ab \mid Ab$

$A \to Sa \mid b$ ⇒

1) $S \to ab \mid Ab$

2) $A \to Sa \mid b$

$A \to abA' \mid bA'$

$A \to aba \mid Aba \mid b$

$A \to Abaa \mid aba \mid b$

$A \to abA' \mid bA'$

$A' \to baA' \mid \in$

$S \to ab \mid Ab$

$A \to abA' \mid bA'$

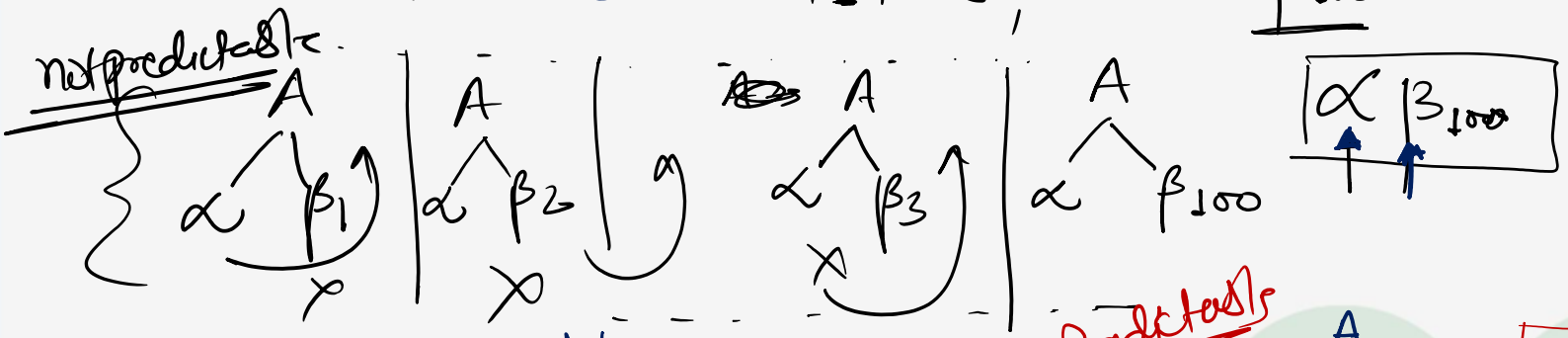
$A' \to baA' \mid \in$

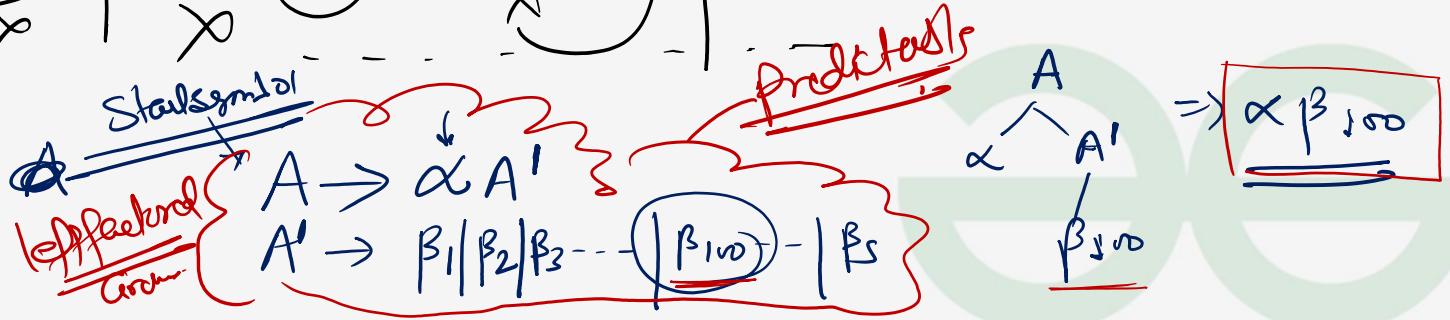
Left factoring and left factored grammar

Eliminate Common prefix:

UnAmbiguous CFG → LeftFactoring Algo → Left-factored Grammar

$G = \{ A \to \alpha\beta_1 \mid \alpha\beta_2 \mid \alpha\beta_3 \mid \cdots \mid \alpha\beta_n \}$, $\omega = \alpha\beta_{100}$





Compiler Design

$G = \{ A \to \alpha\beta_1 \mid \alpha\beta_2 \mid \alpha\beta_3 \mid \dots \mid \alpha\beta_n \}$ ✗

⇓ Remove left factoring

left factored Grammar

$A \to \alpha A'$

$A' \to \beta_1 \mid \beta_2 \mid \beta_3 \mid \dots \mid \beta_n$

Ex: $G_1 = \{ S \to aS \mid ab \mid a \mid \epsilon \}$

⇓

$S \to aS' \mid \epsilon$

$S' \to S \mid b \mid \epsilon$

left factored Gramm

$G_2 = \{ S \to aAb \mid aAa \mid aS \mid b,\ A \to aA \mid b \}$

⇓

$S \to aS' \mid b$

$S' \to Ab \mid Aa \mid S$

$A \to aA \mid b$

⇒

$S \to aS' \mid b$

$S'' \to AS'' \mid S$

$S'' \to b \mid a$

$A \to aA \mid b$

left factored Grammar

Compiler Design

Q-1) G

$S \rightarrow Aa \mid bcA$

$A \rightarrow bd \mid e$

G'

$S \rightarrow bda \mid ea \mid bcA$

$A \rightarrow bd \mid e$

$S \rightarrow bS' \mid ea$

$S' \rightarrow da \mid cA$

$A \rightarrow bd \mid e$

leftfactored Grammar.

Q-2) $S \rightarrow iEtS \mid iEtSeS \mid a$

$E \rightarrow b$

$S \rightarrow iEtSS' \mid a$

$S' \rightarrow eS \mid \epsilon$

$E \rightarrow b$

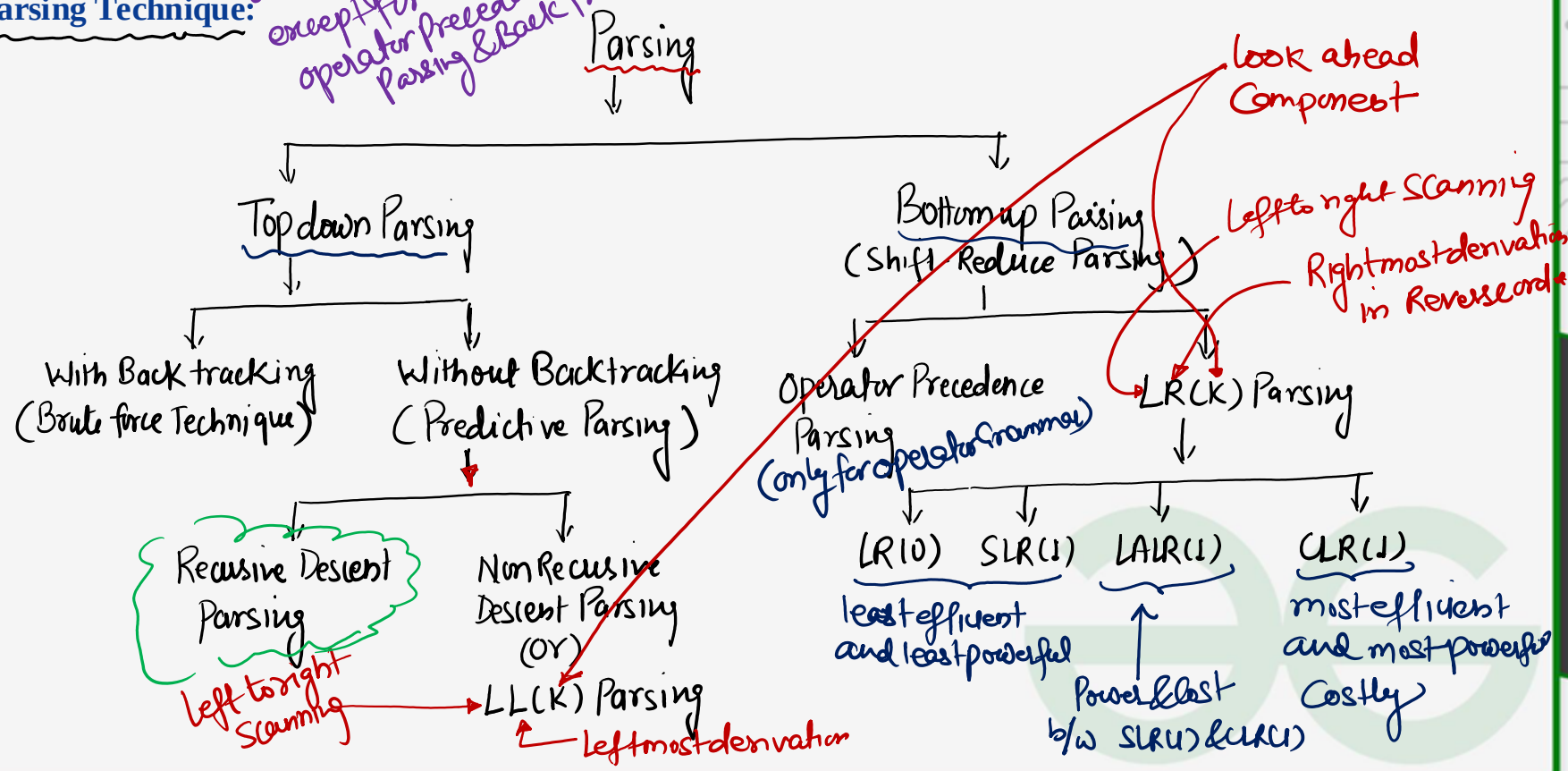
leftfactored Grammar

Compiler Design

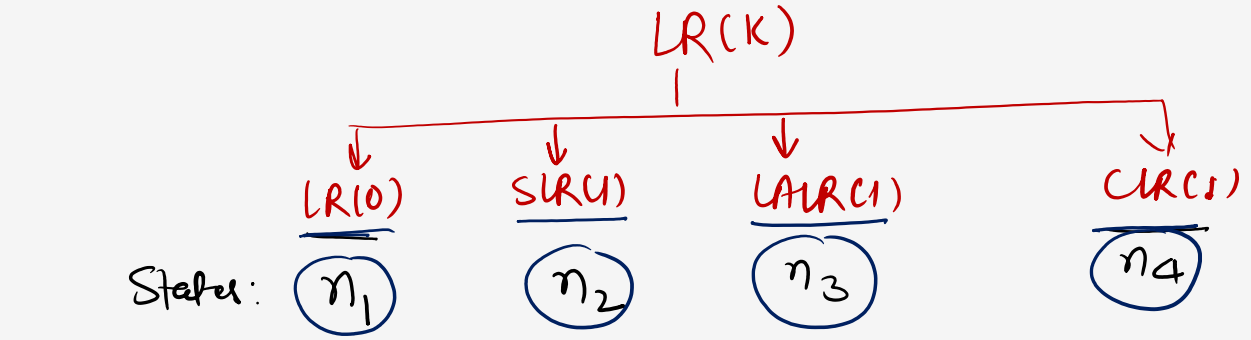
Compiler Design

Parsing Technique:

Grammar should be Unambiguous, except for operator Precedence Parsing & Back Track Parsing



Compiler Design



Note:

1) $n_1 = n_2 = n_3 \leq n_4$

2) No. of Shift entries of in $LR(0) = SLR(1) = LALR(1) \leq CLR(1)$

3) No of Reduce entries $LR(0) \geq SLR(1) \geq LALR(1) \geq CLR(1)$

4) Power of $LR(0) < SLR(1) < LALR(1) < CLR(1)$

Compiler Design

(LL(k))

Top down Passing:

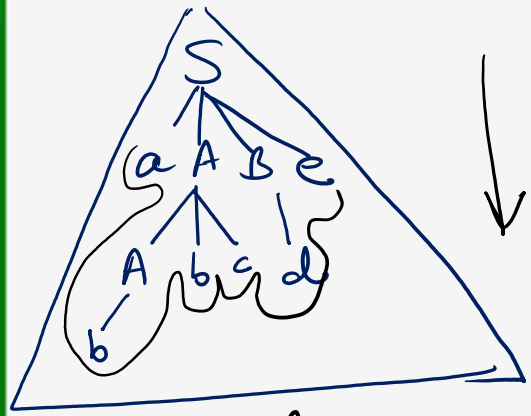
Ex:- $S \to aABe$

$A \to Abc \mid b$

$B \to d$

L → R

$\omega = abbcde$



abbcde

(LR(k))

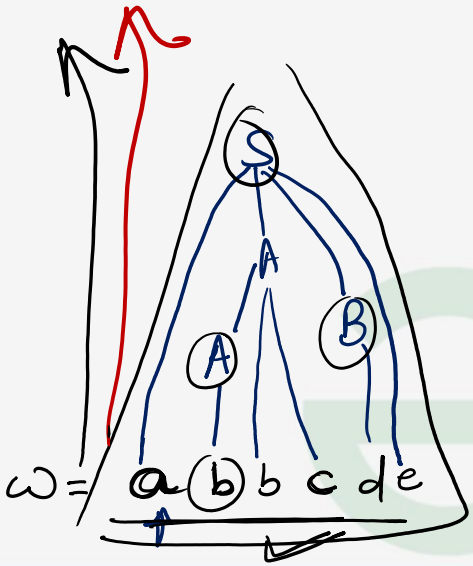
Bottomup passing

Ex:

$S \to aABe$

$A \to Abc \mid b$

$B \to d$



Rightmost derivation in Reverse order

$S \to aABe$

$B \to d$

$A \to Abc$

$A \to b$

Compiler Design

First and Follow Function:

First Function: First(A) is the set of terminals that are the first symbols on the right side of the arrow in every production of A

$A \rightarrow a - | b - | \epsilon - | \epsilon -$

$First(A) = \{a, b, c, \epsilon\}$

Ex: $A \rightarrow aA | bb | cB$

$First(A) = \{a, b, c\}$

$A \rightarrow aA | \epsilon$

$First(A) = \{a, \epsilon\}$

$A \rightarrow aA | BC$

$First(A) = \{a, ?\}$

Ex: $A \rightarrow BC, B \rightarrow bB | aC$

$First(A) = First(B)$ iff $First(B)$ doesnot Contain ϵ

$First(A) = \{a, b\}$, $First(B) = \{a, b\}$

Compiler Design

Procedure:

1. First(a) = {a}
2. if A → a ⇒ First(A) = {a}
3. if A → ∈ ⇒ First(A) = {∈}
4. if A → BC ⇒ First(A) = First(B) iff First(B) does not contain ∈
5. if A → BC ⇒ First(A) = (First(B) − {∈}) ∪ First(C)

First(abc) = {a}

First(a) = {a}

A → BC

First(A) =

A → BC

A → BC

A → BCDE

Compiler Design

Ex: Find First Set

1) G: { S → ABC, A → a, B → bB | a, C → ∈ }

First(S) = {a}

first(A) = {a}

first(B) = {b, a}

first(C) = {e}

2) G: { S → ABC, A → aA | ∈, B → bB | a, C → aC | a }

S → aBC

First(S) = {a, b}

first(A) = {a, ∈}

first(B) = {b, a}

first(C) = {a}

⇒

Compiler Design

Ex1- $G = \{S \to ABC, A \to aA \mid B, B \to bB \mid \epsilon, C \to aC \mid b \mid \epsilon\}$

$First(S) = \{a, b, \epsilon\} \cup \{b, \epsilon\} \cup \{c, b, \epsilon\} = \{a, b, c, \epsilon\}$

$First(A) = \{a, b, \epsilon\}$

$First(B) = \{b, \epsilon\}$

$First(C) = \{b, c, \epsilon\}$

$S \to ABC$

$A \to \epsilon$

$A \to aA \mid B$

$\{a, b, \epsilon\}$

$S \to BC$

$B \to bB$

$B \to Ca$

$C \to aA$

$C \to a$

Ex2 $G = \{S \to aA \mid BC, A \to Aa \mid \epsilon, B \to Ca \mid bB, C \to cA \mid \epsilon\}$

$First(S) = \{a, b, c\}$

$First(A) = \{\epsilon, a\}$

$First(B) = \{b, c, a\} = \{a, b, c\}$

$First(C) = \{c, \epsilon\}$

Compiler Design

Ex:- $G = \{ S \to ACB \mid CbA \mid Ba,\ A \to da \mid BC,\ B \to g \mid \epsilon,\ C \to h \mid \epsilon \}$

$First(S) = \{d, g, h, \epsilon, b, a\} = \{d, g, h, \epsilon, b, a\}$

$First(A) = \{d, g, h, \epsilon\}$

$First(B) = \{g, \epsilon\}$

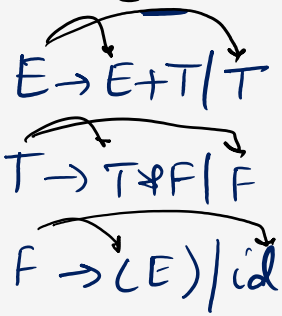
$First(C) = \{h, \epsilon\}$

H.W

Ex:- $G = \{ S \to AaB \mid bC,\ A \to BC \mid bA,\ B \to aB \mid \epsilon,\ C \to SA \mid a \}$

Compiler Design

Ex:- $G = \{ E \to E+T \mid T,\ T \to T*F \mid F,\ F \to (E) \mid id \}$



$first(E) = first(T) = first(F) = \{ (, id \}$

∎

Compiler Design

Follow Function: Follow(A) is the set of terminals that present immediately to the right side of A wherever A is on the RHS of the arrow

Procedure:

1- follow(S) = {$} if S is the start symbol (Start Symbol)

2- if $A \rightarrow \alpha B \beta \Rightarrow follow(B) = first(\beta)$ iff First(β) does not contain $\in$

3- if $A \rightarrow \alpha B \beta \Rightarrow follow(B) = (first(\beta) - \{\in\}) \cup follow(A)$

4- if $A \rightarrow \alpha B \Rightarrow follow(B) = follow(A)$

$first(\beta) = \{x, y\}$

$A \rightarrow \alpha B$

$first(\beta) = \{x, y, \in\}$

Compiler Design

Ex:- Find the follow of each non terminal of the following Grammar

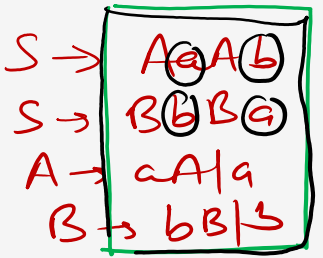
$G = \{ S \to AaAb \mid BbBa,\ A \to aA \mid a,\ B \to bB \mid b \}$

$Follow(S) = \{\$\}$

$Follow(A) = \{a, b\}$

$Follow(B) = \{a, b\}$

Any



Ex:- $G = \{ S \to AaBb \mid AC,\ A \to aB \mid b,\ B \to a,\ C \to c \}$

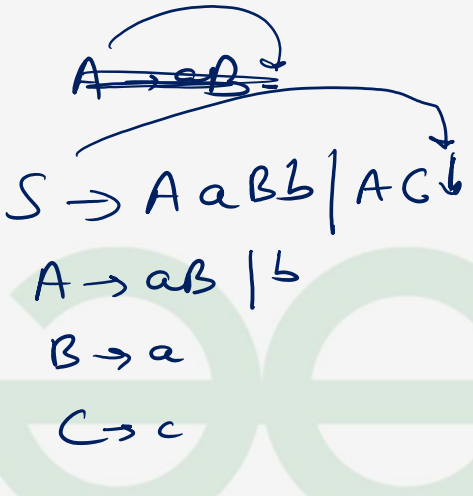
$Follow(S) = \{\$\}$

$Follow(A) = \{a, c\}$

$follow(B) = \{b, follow(A)\} = \{b, a, c\}$

$follow(C) = \{follow(S)\} = \{\$\}$

Any



Compiler Design

Ex:- $G = \{ S \to ABC, A \to aA/\epsilon, B \to bB/\epsilon, C \to cC/\epsilon \}$

Soln:

$follow(S) = \{\$\}$

$follow(A) = \{b, c, follow(S)\} = \{b, c, \$\}$

$follow(B) = \{c, follow(S)\} = \{c, \$\}$

$follow(C) = \{follow(S)\} = \{\$\}$

Ans

follow(A):- first( )

$S \to A$

$S \to AB$

Compiler Design

H.W

Ex: $G = \{ S \to SAB \mid \in,\ A \to AB \mid \in,\ B \to BC \mid Ca,\ C \to aCb \mid a$

Compiler Design

H.W

Ex:- $G = \{S \to ACB \mid CbB \mid Ba, A \to da \mid BC, B \to c \mid \epsilon, C \to e \mid \epsilon\}$

Compiler Design

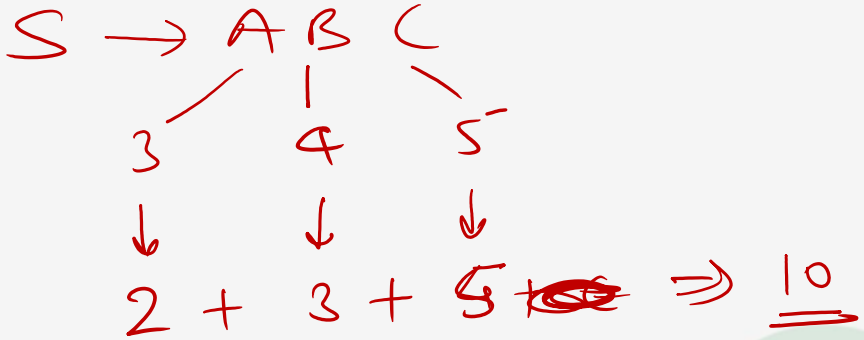
H.W

Ex:- $G = \{E \to E+T/T,\ T \to T*F/F,\ F \to (E)/id\}$

Compiler Design

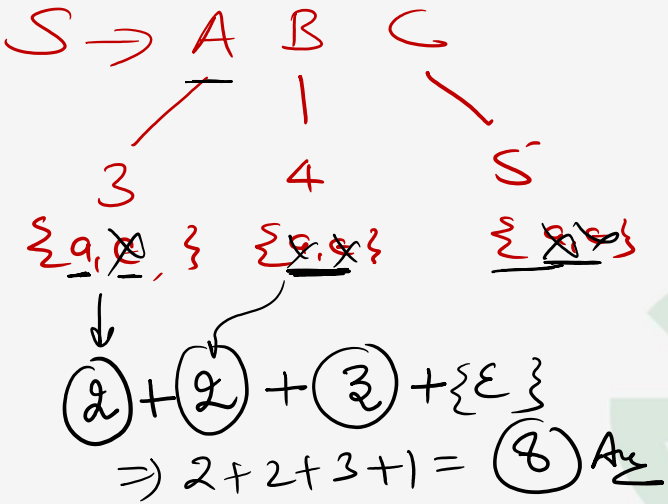
Compiler Design

Q. Consider the following Production $S \rightarrow ABC$, of the no. of elements in First(A), First(B), and First(C) are 3,4,5 elements respectively and all these First sets contain 'ϵ' and the remaining all symbols are different. Then find no. of elements in First(S)?



Compiler Design

Question: Consider the following Production $S \rightarrow ABC$, if the no. of elements in First(A), First(B) and First(C) are 3, 4, 5 elements respectively and all these First sets Contains $\{a, \epsilon\}$ and the remaining all symbols are different. Then find no. of elements in First(S)?



Thank You !