MODULE :2
Stacks and Queues

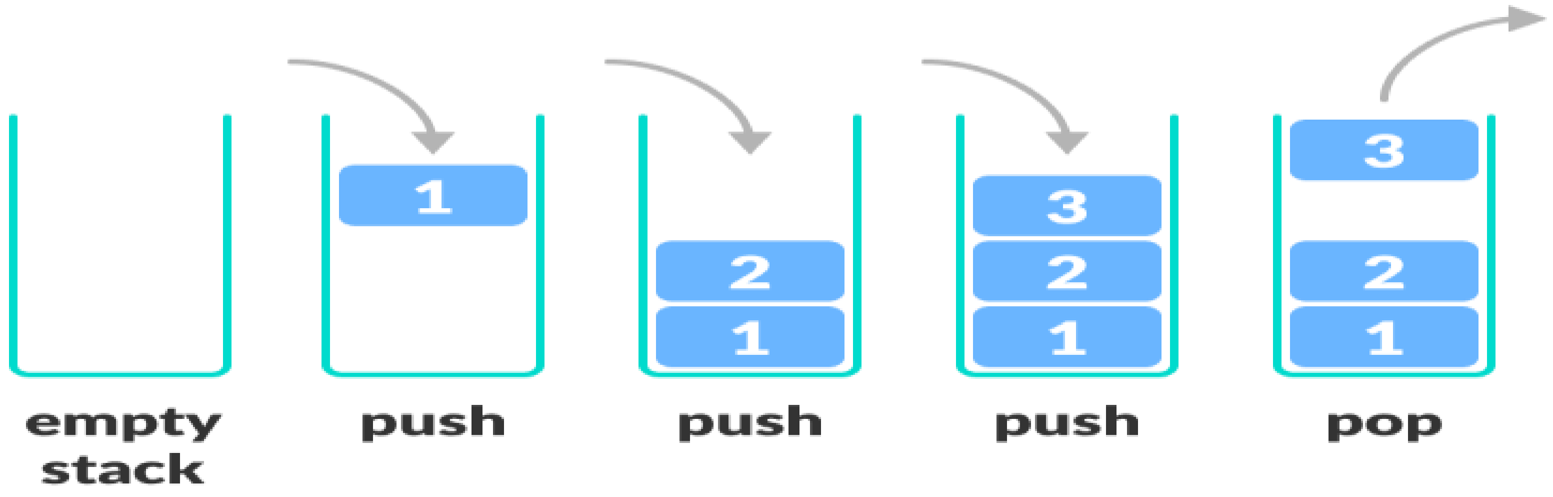
Contents

Abstract Data Types (ADTs)

- **Abstraction**? Anything that hides details & provides only the essentials.
- **Abstract Data Types (ADTs):** Simple or structured data types whose implementation details are hidden…

Stack Definition

- It is a list of elements in which insertion and Deletion can takes place at one end only i.e. TOP end of Stack.



Stack

- A stack is a data structure that stores data in such a way that the last piece of data stored, is the first one retrieved
  - – also called last-in, first-out
- Only access to the stack is the top element

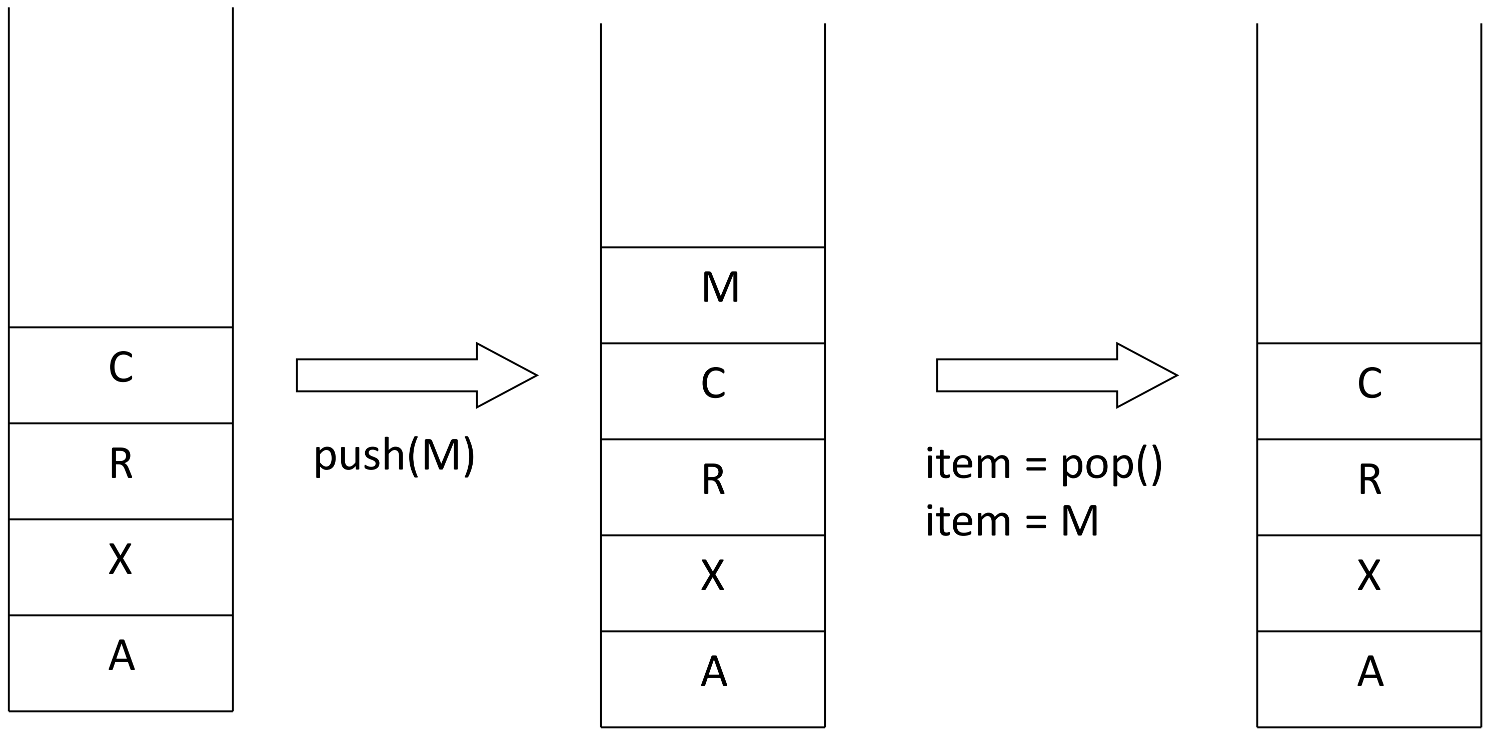
Eg. consider trays in a cafeteria

- to get the bottom tray out, you must first remove all of the elements above

Operations associated with Stack

- *Push*
  - the operation to place a new item at the top of the stack
- *Pop*
  - the operation to remove the next item from the top of the stack

Push and Pop Operation in Stack



Important Terms

- **OVERFLOW**: new data is to be inserted into data structure but there is no available space.
- **UNDERFLOW:** situation where one wants to delete data from data structure which is empty.

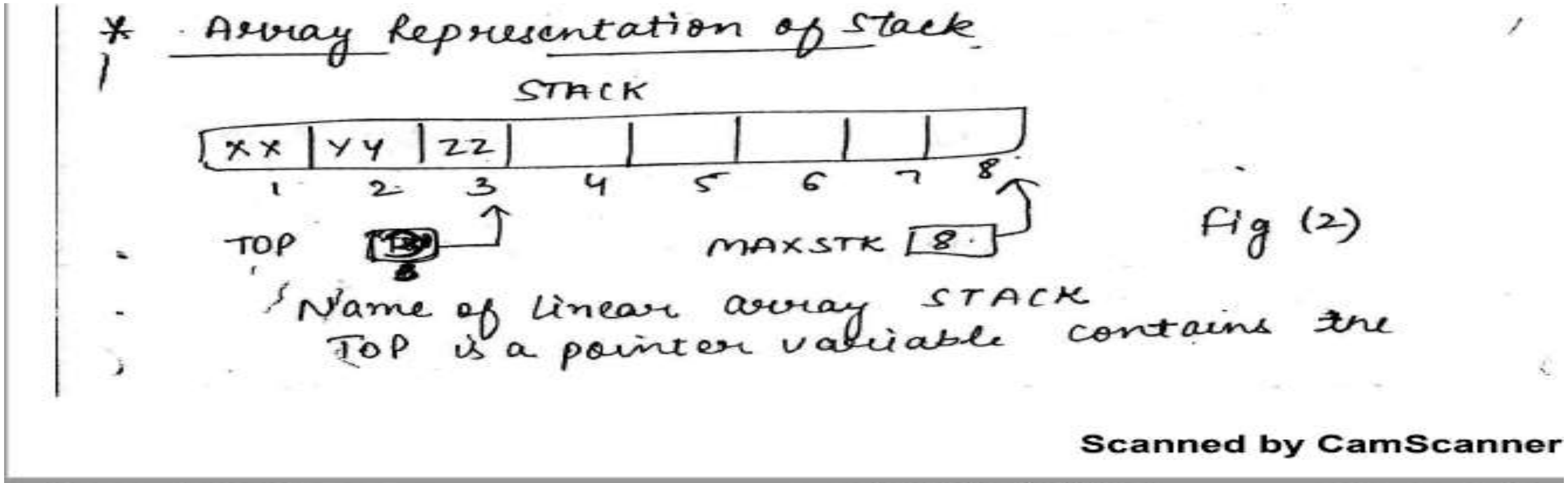
Stack Operations:

- **Push:** Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
- **Pop:** Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
- **Peek or Top:** Returns top element of stack.
- **isEmpty:** Returns true if stack is empty, else false.

Implementing a Stack

- Two different ways to implement a stack
  - array
  - linked list

Array Representation Of Stack



* Array Representation of Stack

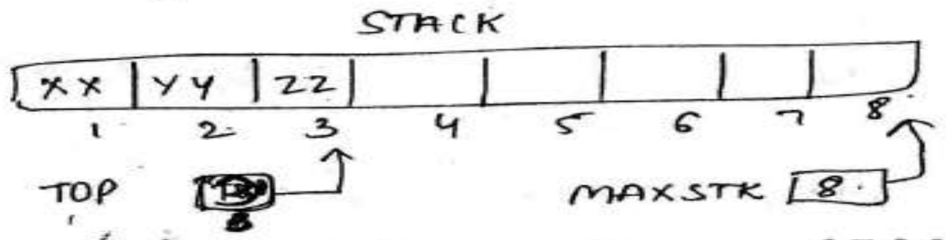
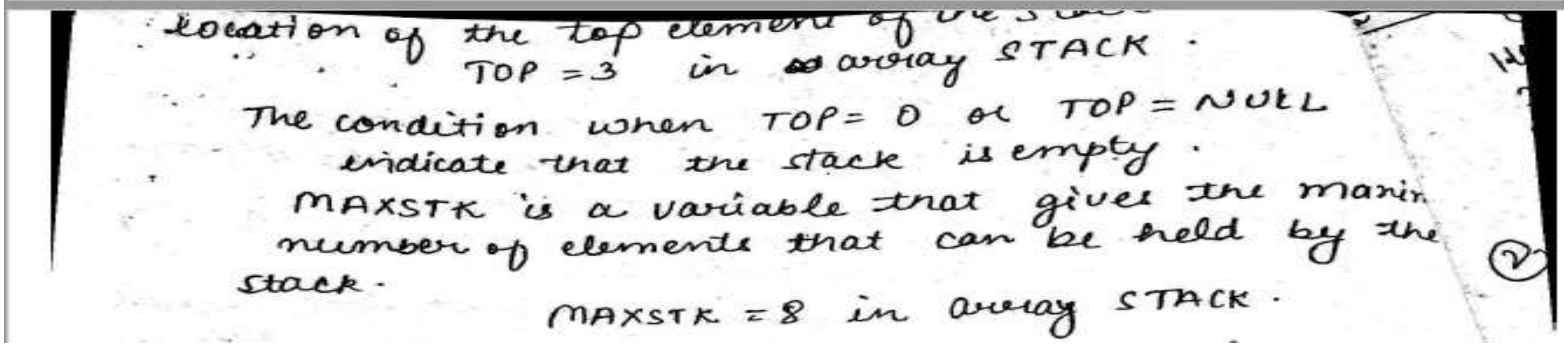


Fig (2)

Name of linear array STACK
TOP is a pointer variable contains the



location of the top element of the
∴ TOP = 3 in array STACK.

The condition when TOP = 0 or TOP = NULL indicate that the stack is empty.

MAXSTK is a variable that gives the maxim number of elements that can be held by the stack.

MAXSTK = 8 in array STACK.

Overflow and underflow conditions:

- OVERFLOW= when TOP= MAXSTK
- UNDERFLOW= when TOP=0

Implementing Stacks: Array

- Advantages
  - best performance
- Disadvantage
  - fixed size
- Basic implementation
  - initially empty array
  - field to record where the next data gets placed into
  - if array is full, push() returns false
    - otherwise adds it into the correct spot
  - if array is empty, pop() returns null
    - otherwise removes the next item in the stack

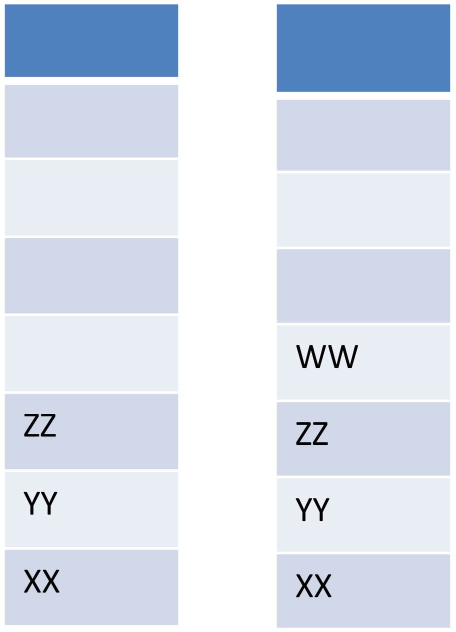
Push Operation in Stack

- **ALGORITHM:-**
- **PUSH(STACK,TOP,MAXSTK,ITEM)**
- **This algorithm insert an element i.e ITEM into a STACK.**
- **[STACK already filled ?]if TOP:=MAXSTK then write: OVERFLOW & Exit**
- **Set TOP:=TOP+1 [Increment TOP By 1]**
- **Set STACK [TOP]:=ITEM [ITEM insert onto STACK]**
- **Exit**

Push Operation Example:

PUSH(STACK, WW)

1. Let TOP= 3
2. Set TOP= TOP+1 = 3+1=4
3. STACK[TOP]= STACK[4]=WW
4. Return



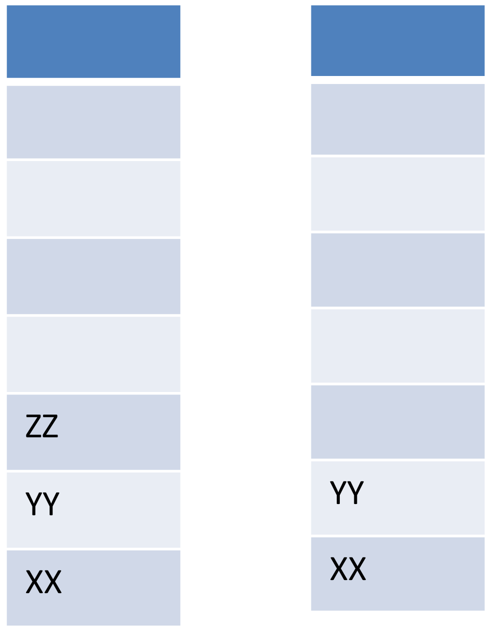
Pop Operations in Stack

- **POP(STACK,TOP,ITEM)**
- **This procedure delete the TOP element of STACK & assign to the variable ITEM.**
- **[STACK has an ITEM to be removed?]**
- **If TOP:=0 then Print: UNDERFLOW & Exit**
- **Set ITEM:=STACK[TOP]   [Assign TOP element to ITEM]**
- **Set TOP:=TOP-1[Decrement TOP by 1]**
- **Exit**

POP Operation Example:

POP(STACK, ITEM)

1. Let TOP= 3
2. ITEM=ZZ
3. TOP=3-1=2
4. Return



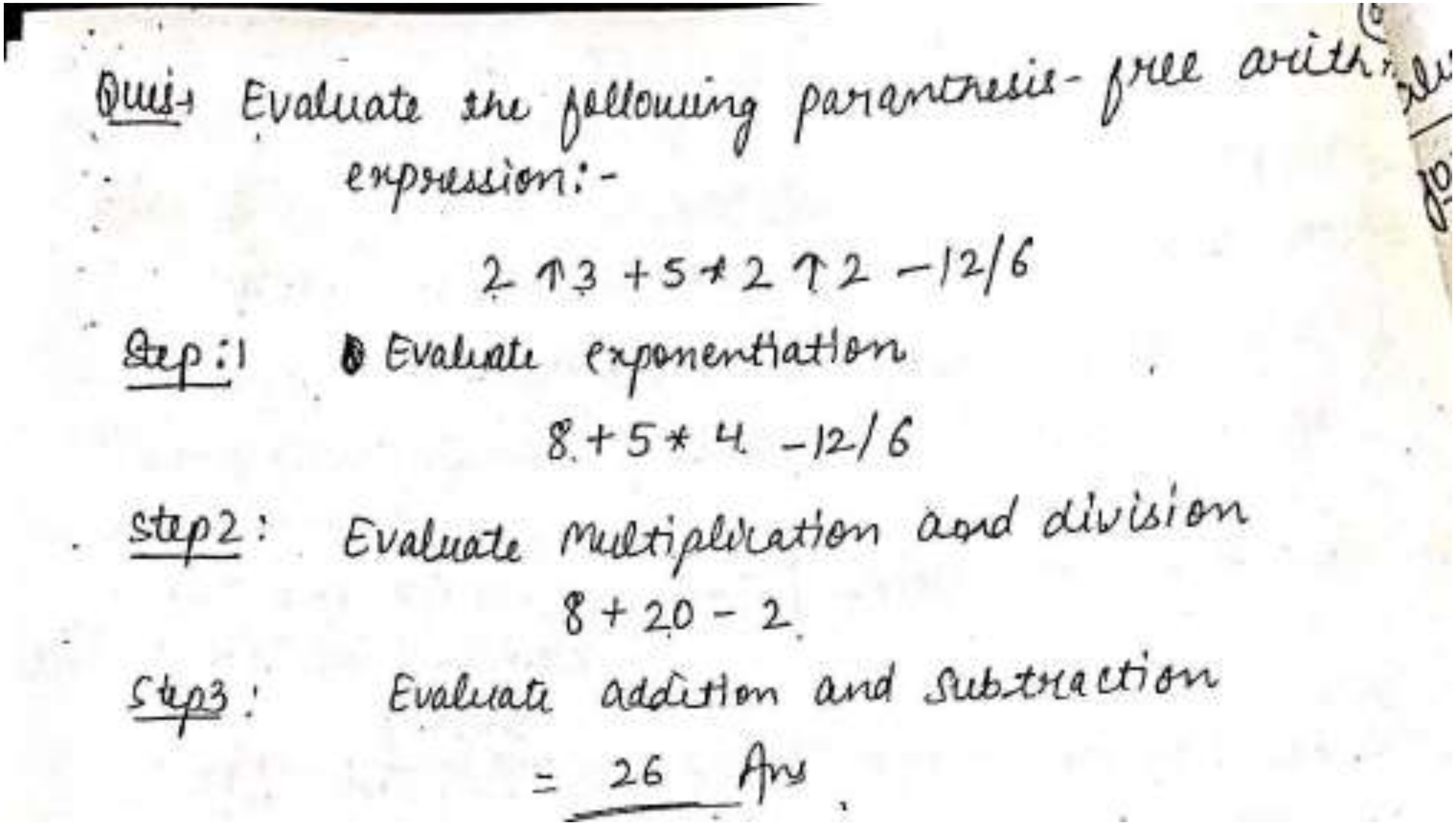
Implementing a Stack: Linked List

- Advantages:
  - always constant time to push or pop an element
  - can grow to an infinite size
- Disadvantages
  - the common case is the slowest of all the implementations
- Basic implementation
  - list is initially empty
  - *push()* method adds a new item to the head of the list
  - *pop()* method removes the head of the list

ARITHMETIC EXPRESSIONS Levels of precedence

- Highest: Exponentiation (^or $\uparrow$)
- Next highest: Multiplication (*) and division (/)
- Lowest: Addition (+) and Subtraction (-)

EXAMPLE:



Ques:- Evaluate the following paranthesis- free arith

expression:-

2 ↑ 3 + 5 * 2 ↑ 2 − 12/6

Step:1 Evaluate exponentiation

8 + 5 * 4 − 12/6

step2: Evaluate multiplication and division

8 + 20 − 2

Step3: Evaluate addition and subtraction

= 26 Ans

Arithmetic Notations

- **Polish notation (prefix notation) –**
  It refers to the notation in which the operator is placed before its two operands .

Eg: +AB , - CD,*EF, /GH

- **Reverse Polish notation(postfix notation) –**
  It refers to the analogous notation in which the operator is placed after its two operands.

- Eg: AB+, CD- ,EF* , GH/

Arithmetic Notations contd..

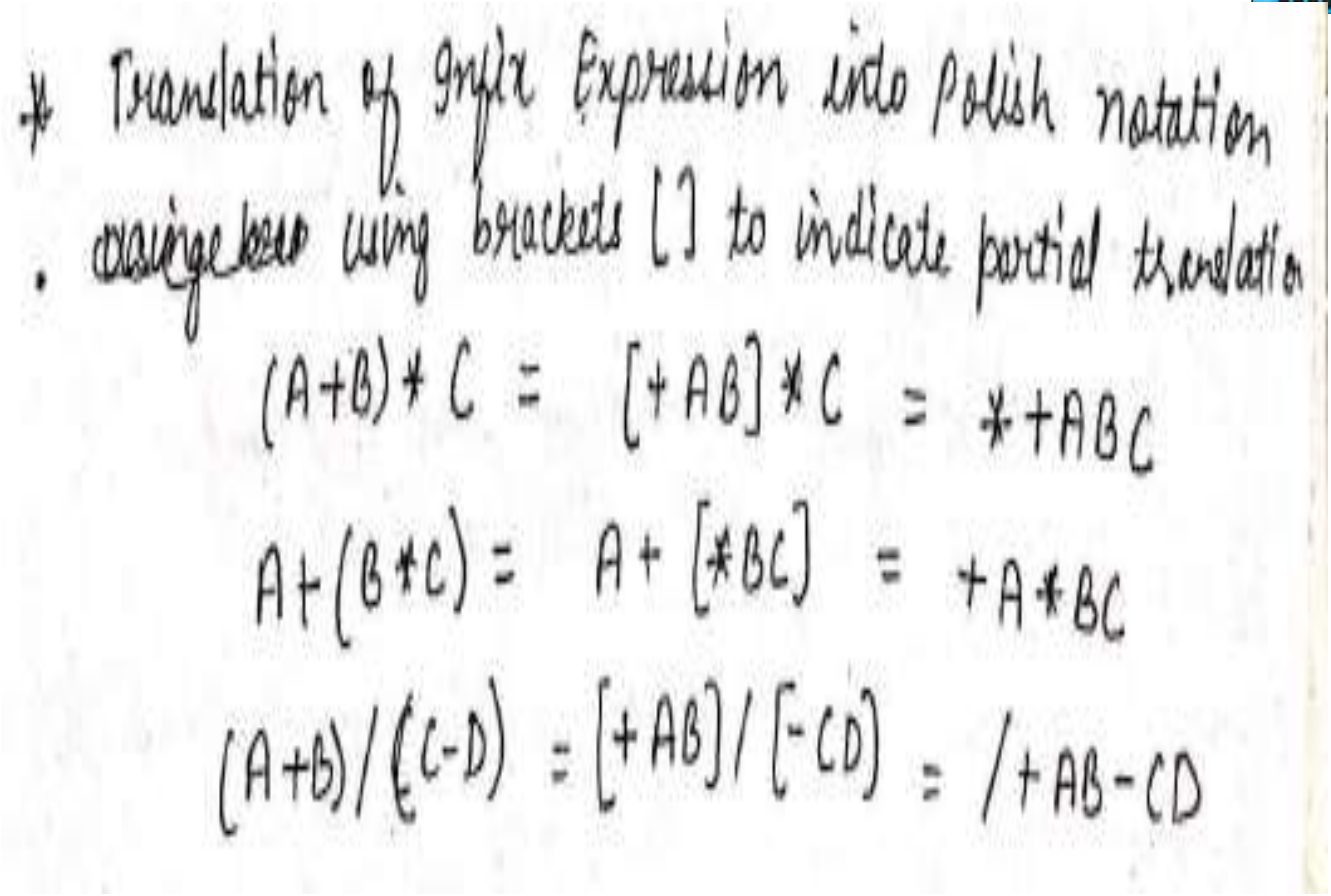
- ***Infix Notation****:* Operators are written between the operands they operate on.

Eg: A+B , C-D, E*F, G/H

Convert Infix To Prefix Notation

- Input : A * B + C / D
- Output : + * A B/ C D

- Input : (A - B/C) * (A/K-L)
- Output : *-A/BC-/AKL



* Translation of Infix Expression into Polish notation

• [illegible] using brackets [ ] to indicate partial translation

$$(A+B)*C = [+AB]*C = *+ABC$$

$$A+(B*C) = A+[*BC] = +A*BC$$

$$(A+B)/(C-D) = [+AB]/[-CD] = /+AB-CD$$