Arithmetic and Logical Expressions

THE POLISH NOTATION

Dr.Sudhirkumar J. Patel

Arithmetic and Logical Expressions

- repeatedly scan through the expression
- take parentheses and priorities of operators into account

```
a + b + c * d - e / g
a + b + ( c * d ) - ( e / g )
a + (( b + c ) * d - e ) / g

a + b <= c && a + b <= d
( a + b <= c ) || ( a + b <= d )
```

The Polish Notations

Q : How can a compiler accept an expression and produce correct code ?
A : Tranforming the expression into a form called Polish notation

Infix form	Prefix form	Postfix form
a * b	* a b	a b *
a + b * c	+ a * b c	a b c * +
(a + b) * c	* + a b c	a b + c *

Reverse Polish
notation

Expression Evaluations : Stacks

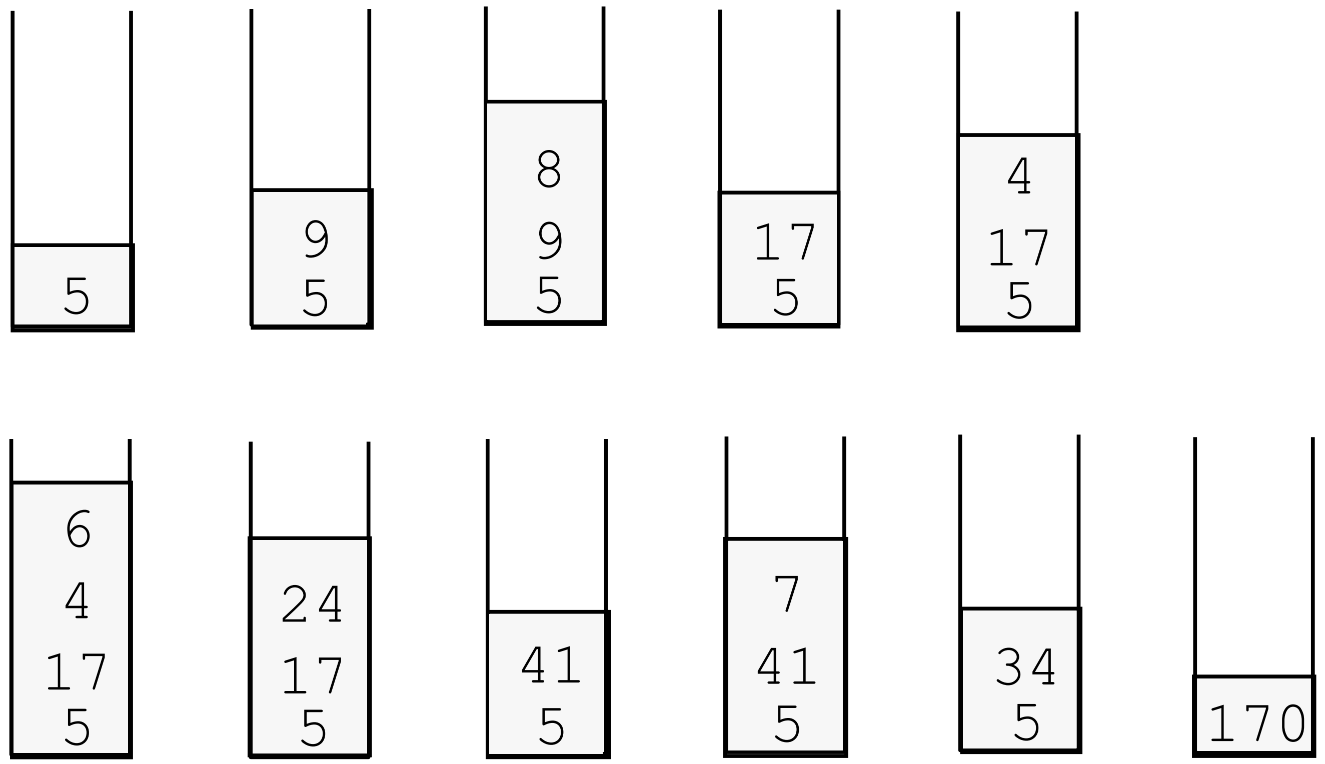
```
5 * ( ( ( 9 + 8 ) + ( 4 * 6 ) ) - 7 )
Postfix form : 5 9 8 + 4 6 * + 7 - *

Push( 5 )
Push( 9 )
Push( 8 )
Push( Pop() + Pop() )
Push( 4 )
Push( 6 )
Push( Pop() * Pop() )
Push( Pop() + Pop() )
Push( 7 )
Push( Pop() - Pop() )
Push( Pop() * Pop() )
```

Expression Evaluations : Stacks

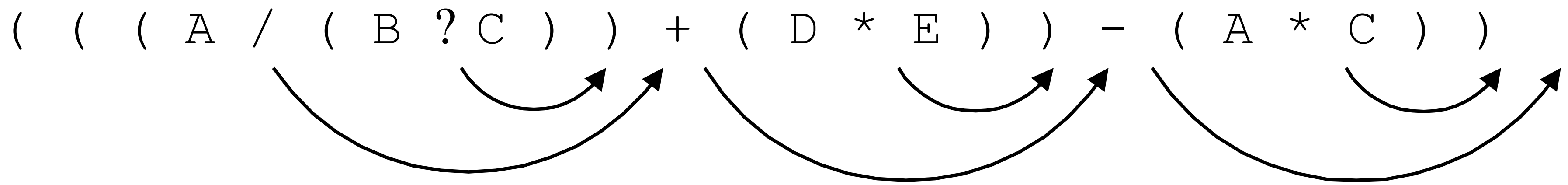
```
5 * ( ( ( 9 + 8 ) + ( 4 * 6 ) ) - 7 )
```

`Postfix form :` ***5 9 8 + 4 6 * + 7 - ****



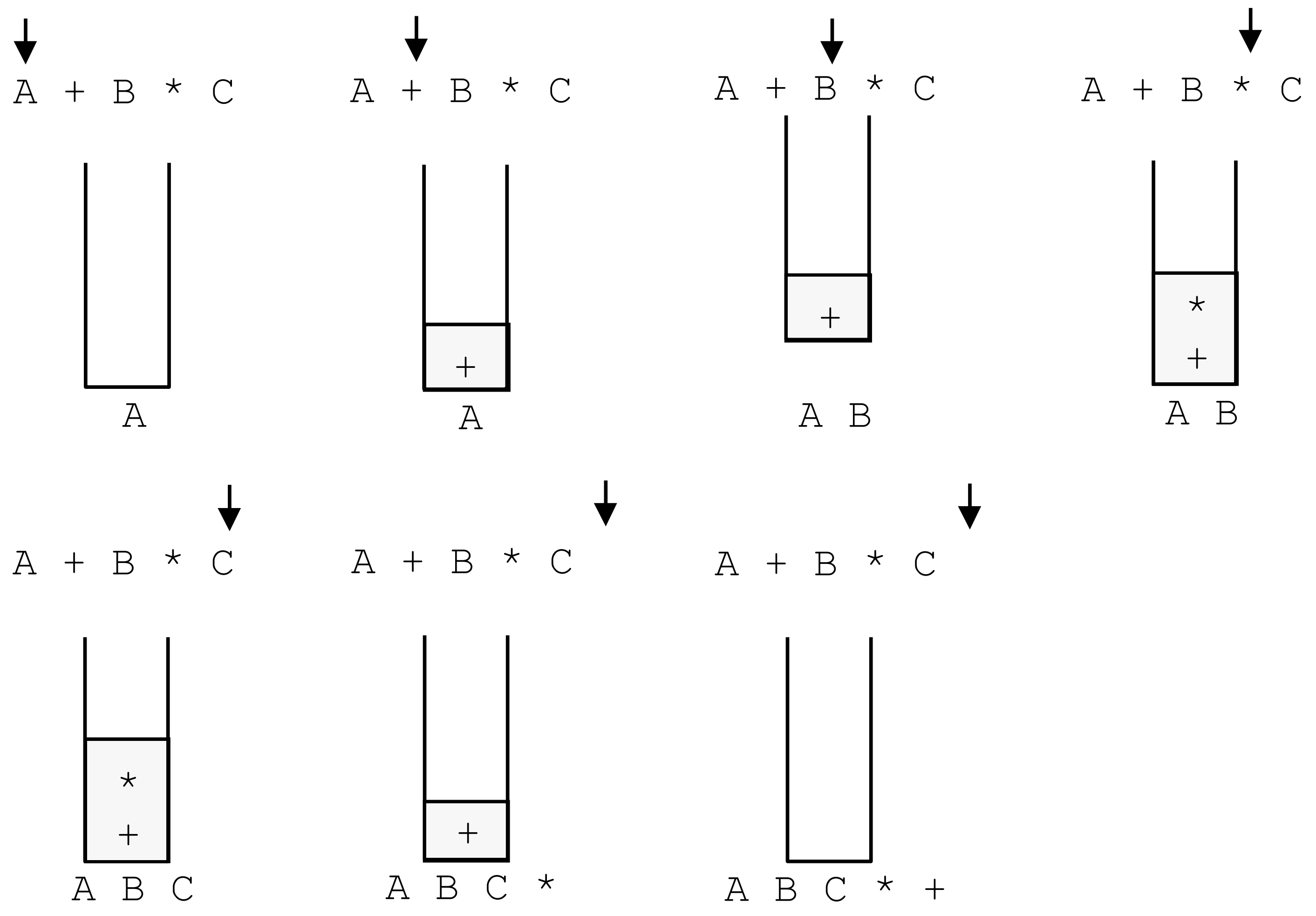
Infix Form → Postfix Form

A / B ? C + D * E - A * C

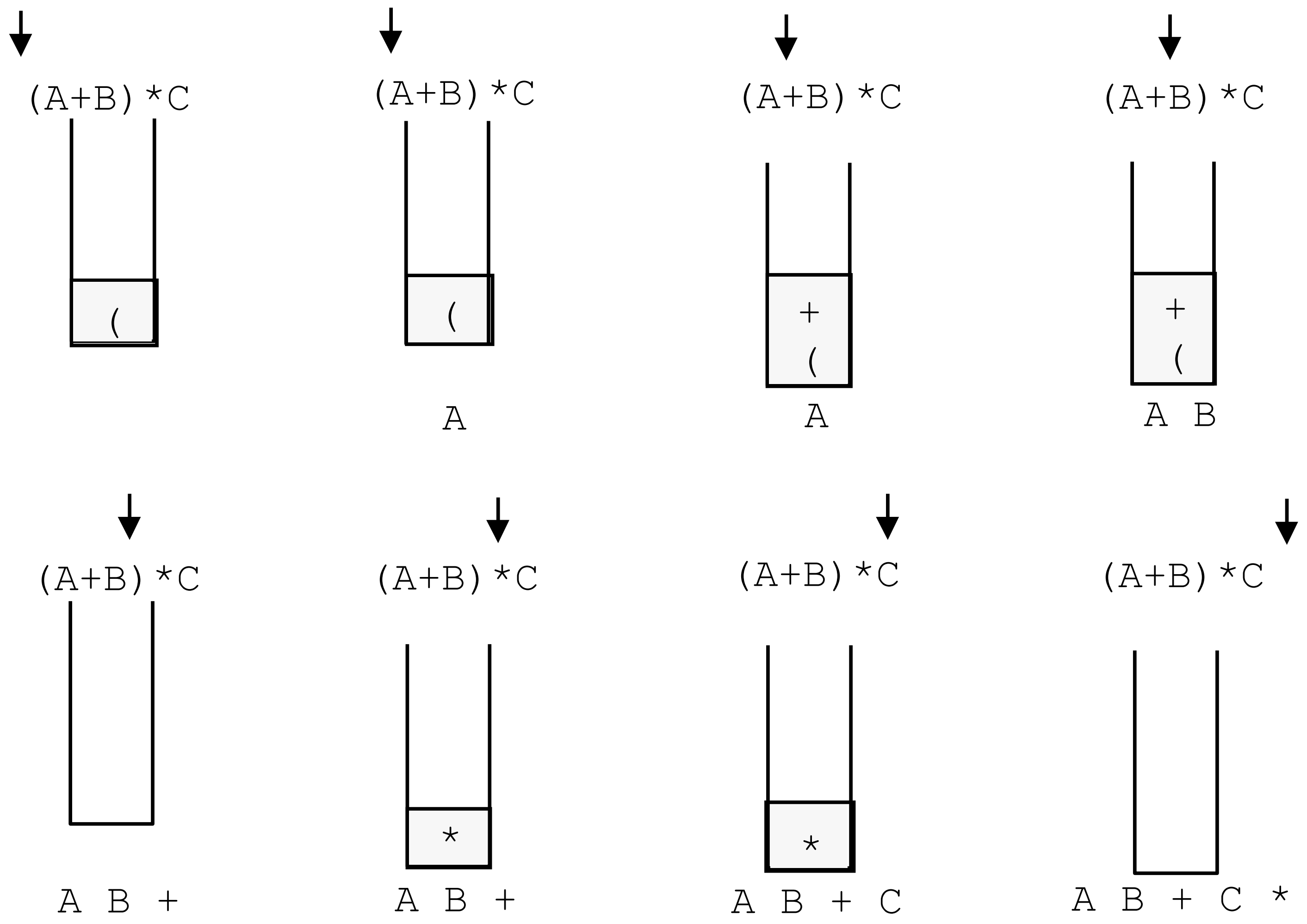


A B C ? / D E * + A C * -

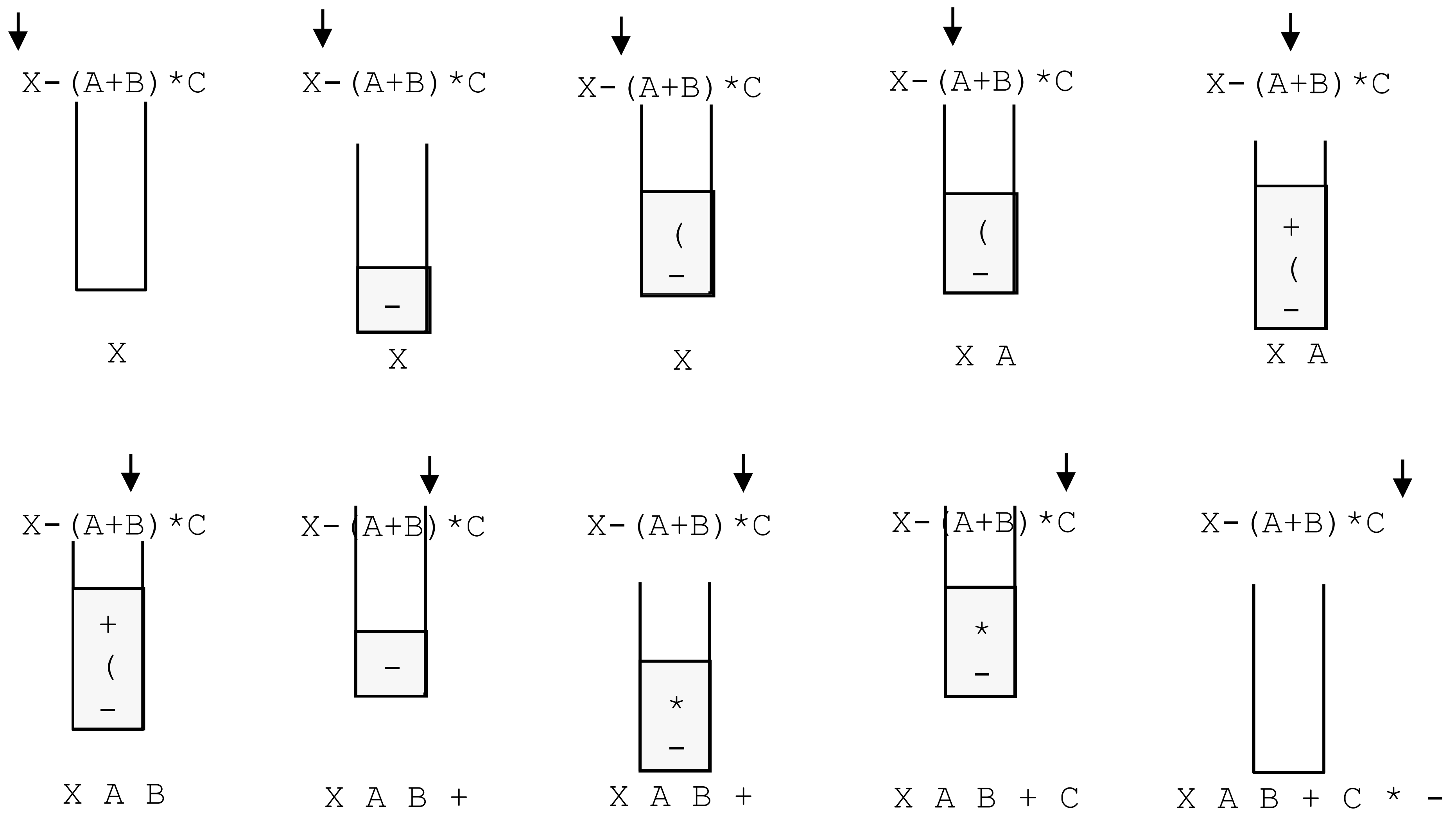
Infix Form $\rightarrow$ Postfix Form



Infix Form → Postfix Form



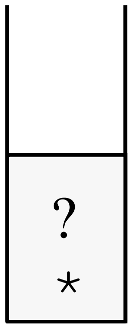
Infix Form $\rightarrow$ Postfix Form



Operator Priorities

Symbol	In-Stack Priority	In-Coming Priority
)	-	-
?	3	4
*, /	2	2
+, -	1	1
(	0	4

Operators are taken out of the stack as long as the in-stack priority is greater than or equal to the in-coming priority of the new operator.



```
                        ↓
input : a*b?2 + 3
output: ab2
      : ab2?*
```