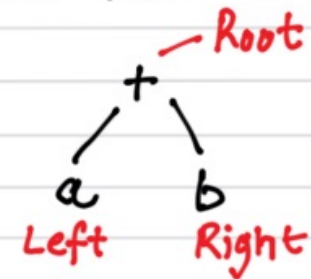


3.4.3

REVERSE POLISH NOTATION (RPN):

A mathematical equation is always INORDER; means, all of the operators are between operands. E.g: $n + e \div m$



- A computer doesn't recognise inorder equations.

- It converts every inorder equation to its equivalent postorder equation.

WRITE

infix

prefix

postfix

NOTATION

$a + b$

$+ab$

$ab+$

READ

inorder

pre order

post order

Left, Root, Right

Root, Left, Right

Left, Right, Root

↓ Reverse Polish Notation (RPN)

- This conversion eliminates the complication in formation of solution order. E.g: in postorder there is no need of BODMAS implementation.

- Conversion from infix to other forms of notations is only possible through BINARY TREE, i.e. firstly BT is made and then read in a different required order...



Zak
ZAFAR ALI KHAN

3.4.3 Translation softwaree.pdf - Adobe Acrobat Pro DC

File Edit View Window Help

Home Tools 9608-syllabus-201... 4.1.2 Algorithms.pd... 3.4.3 Translation so... x

DL: 16.0 kbps UL: 227.3 kbps Sign In

17 / 19

Practical implications:

- Reverse Polish expression can be processed directly from left to right
- RPN is free from ambiguities.
- Does not require brackets: the user simply performs calculations in the order that is required, letting the automatic stack store intermediate results on the fly for later use.
- There is no requirement for the precedent rules required in infix notation.
- Calculations occur as soon as an operator is specified. Thus, expressions are not entered wholesale from left to right but calculated one piece at a time.
- The automatic stack permits the automatic storage of intermediate results for later use. **This key feature is what permits RPN calculators to easily evaluate expression of arbitrary complexity also; they do not have limits on the complexity of expressions they can evaluate.**
- In RPN calculators, no equals key is required to force computation to occur.
- Users must know the size of the stack, because practical implementations of RPN use different sizes for the stack.

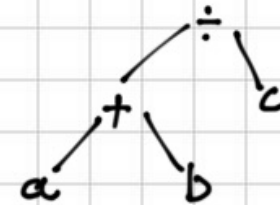
Windows taskbar: 9:43 PM 01/05/2017

Walkthrough
Chart for
infix - postfix
conversion.

- 1/ INFIX
- 2/ BINARY TREE

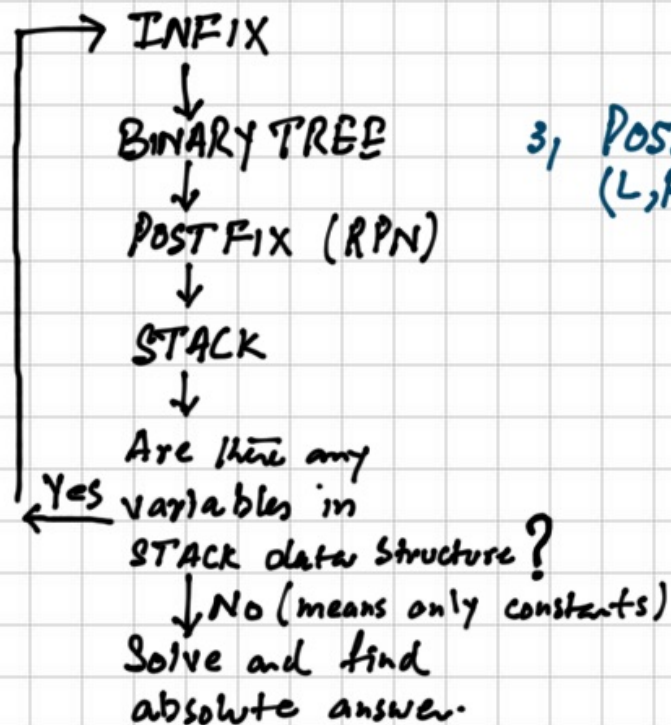
$$(a+b)/c$$

Follow BODMAS



$$ab+c\div \quad (\text{RPN})$$

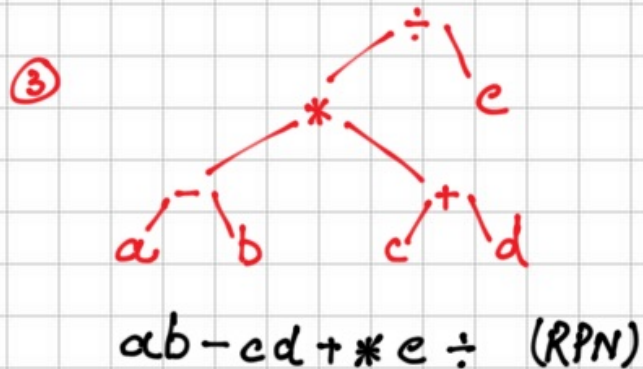
- 3/ POSTFIX
(L, R, Root)



Zak
ZAFAR ALI KHAN

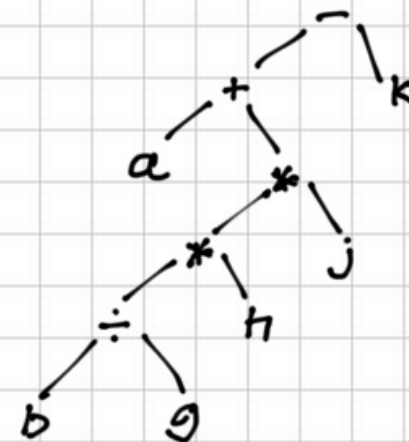
Find RPN of
following infix
notation:

- 1, $a+b/g*h*j-k$
- 2, $((a+b)/((g*h)*(j-k)))$
- 3, $((a-b)*(c+d))/e$
- 4, $abc-d$
- 5, $a+b+c/d$



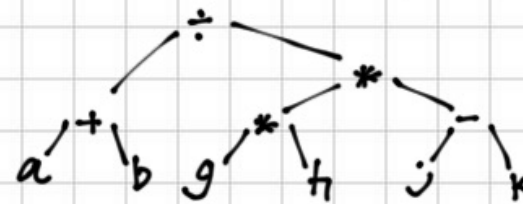
-BODMAS-

1, $a+b/g*h*j-k$



$abg\div h*j*+k-$

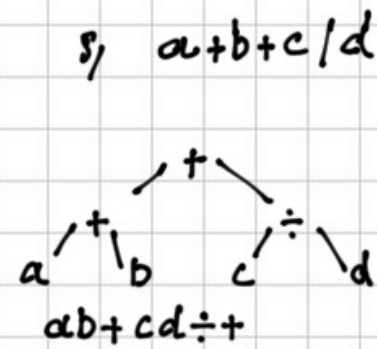
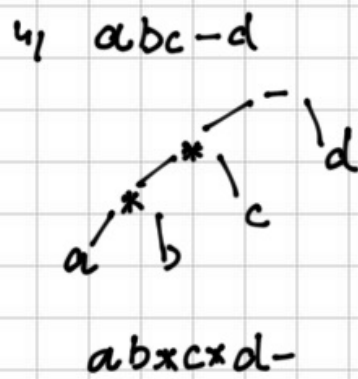
2, $((a+b)/((g*h)*(j-k)))$



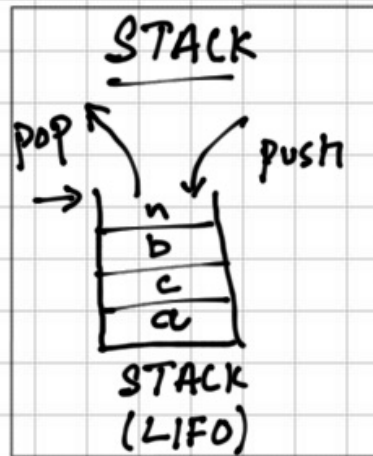
$ab+gh*jk-*\div$



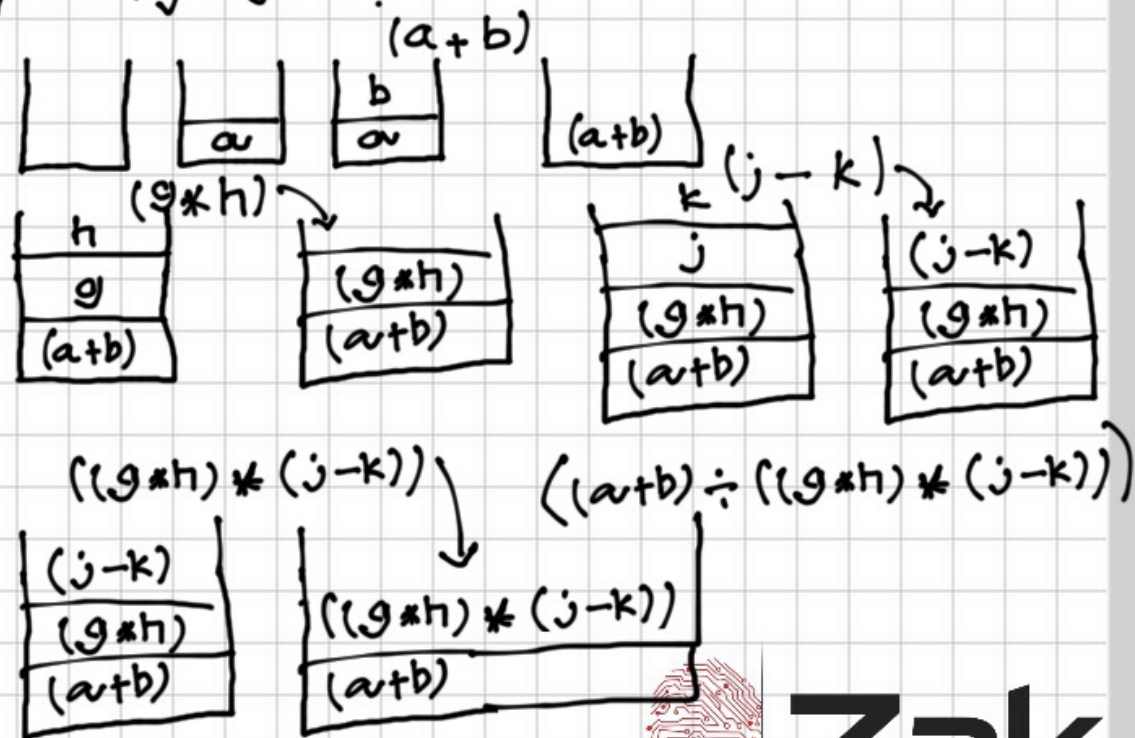
Zak
ZAFAR ALI KHAN



Zak
ZAFAR ALI KHAN



2, $ab+gh*jk-*\div$



Zak
ZAFAR ALI KHAN

1) Find postfix variables
 2) Replace with their values using stack
 3) Find answer


Classwork
 $A - B * C$
 where; $A = 2$
 $B = 3$
 $C = 4$

$((((15 - 10) \div (15 \div 3)) + 9) * 5)$

$15 \ 10 \ - \ 15 \ 3 \ \div \ \div \ 9 \ + \ 5 \ *$

Stack operations for postfix evaluation:

- Initial stack: $\begin{array}{|c|} \hline 10 \\ \hline 15 \\ \hline \end{array}$
- Operation $15 - 10$: $\begin{array}{|c|} \hline 3 \\ \hline 15 \\ \hline 5 \\ \hline \end{array}$
- Operation $15 \div 3$: $\begin{array}{|c|} \hline 5 \\ \hline 5 \\ \hline \end{array}$
- Operation $5 \div 5$: $\begin{array}{|c|} \hline 1 \\ \hline 9 \\ \hline \end{array}$
- Operation $1 + 9$: $\begin{array}{|c|} \hline 10 \\ \hline \end{array}$
- Operation $10 * 5$: $\begin{array}{|c|} \hline 50 \\ \hline \end{array}$


Zak
 ZAFAR ALI KHAN

