

Paste New Slide Reuse Slides Layout Reset Section

Font Paragraph

Text Direction Align Text Convert to SmartArt

Drawing

Editing

Share Comments Dictate Design Ideas Voice Designer

1 Data Structures & Algorithms Stack & its Applications

2 Classification of Data Structures

3 STACK

4 Various Types of Stacks

5 LIFO

Study Point

Data Structures & Algorithms

Stack & its Applications


Presented By:

Sh. Muhammad Aamir
Lecturer
Department of Computer Science
GC University, Faisalabad

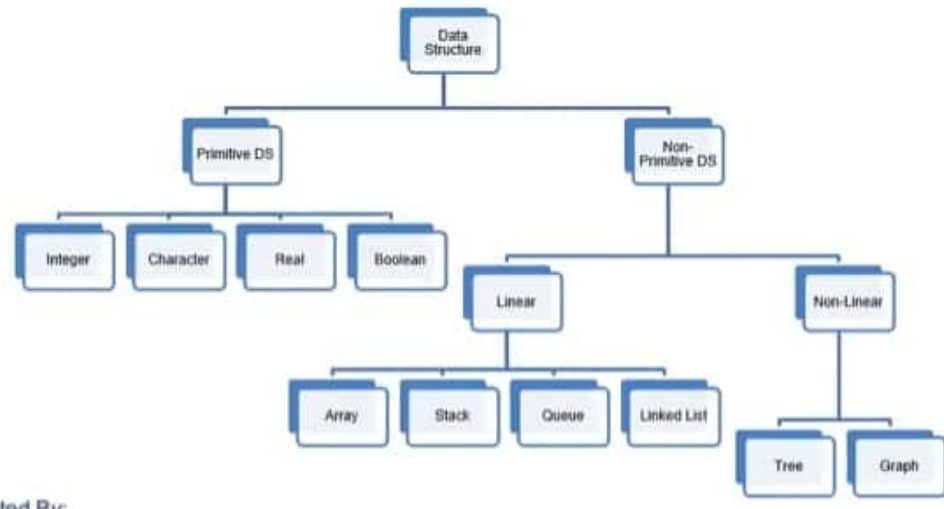
Click to add notes

- 1 Data Structures & Algorithms Stack & Its Applications
- 2 Classification of Data Structures
- 3 STACK
- 4 Various Types of Stacks
- 5 LIFO

2


Study Point

Classification of Data Structures

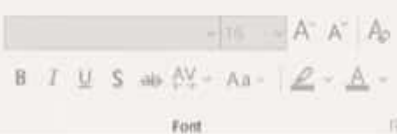


```
graph TD; DS[Data Structure] --> PDS[Primitive DS]; DS --> NPS[Non-Primitive DS]; PDS --> Integer; PDS --> Character; PDS --> Real; PDS --> Boolean; NPS --> Linear; NPS --> NonLinear[Non-Linear]; Linear --> Array; Linear --> Stack; Linear --> Queue; Linear --> LL[Linked List]; NonLinear --> Tree; NonLinear --> Graph
```

Presented By:
Sheikh Muhammad Aamir

Click to add notes

File Home Insert Design Transitions Animations Slide Show Review View Help Foxit Reader PDF

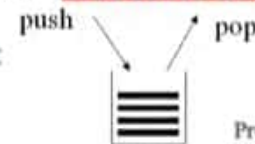


STACK



- A stack is a Non – Primitive Linear Data Structure
- List of elements in which an element is inserted or deleted only at one end called **TOP**
- To add (**push**) an item to the stack, it must be placed on the top of the stack.
- To remove (**pop**) an item from the stack, it must be removed from the top of the stack too.
- Thus, the last element that is pushed into the stack, is the first element to be popped out of the stack i.e. **in reverse order**.

i.e., A stack is a **Last In First Out (LIFO)** data structure



Presented By:
Sheikh Muhammad Aamir

Click to add notes

1 Data Structures & Algorithms Stack & Its Applications

2 Classification of Data Structures

3 STACK


- It is a linear data structure.
- It follows the LIFO (Last In First Out) principle.
- It is used to store data in a sequential manner.
- It is used to store data in a sequential manner.
- It is used to store data in a sequential manner.

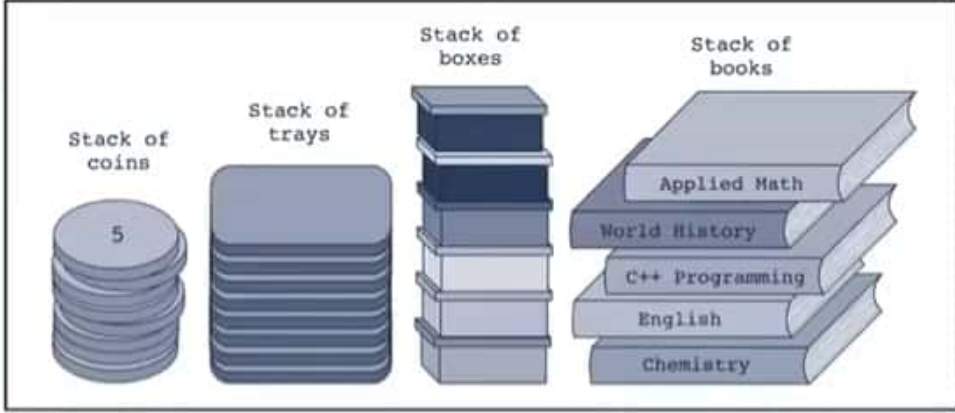
4 Various Types of Stacks

5 LIFO

- Last In First Out (LIFO) data structure

Various Types of Stacks

 Study Point



Stack of coins

Stack of trays

Stack of boxes

Stack of books

- Applied Math
- World History
- C++ Programming
- English
- Chemistry

Presented By:
Sheikh Muhammad Aamir

3 STACK

4 Various Types of Stacks


5 LIFO

6 Postponed Decisions

7 Postponed Decisions

LIFO

- Last In First Out (LIFO) data structure
- Top element of stack is first element to be removed from stack
- Elements added and removed from one end (top)

 Study Point

Presented By:
Sheikh Muhammad Aamir

3 STACK


4 Various Types of Stacks

5 LIFO

6 Postponed Decisions

7 Postponed Decisions...

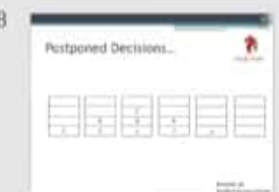
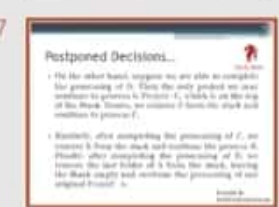
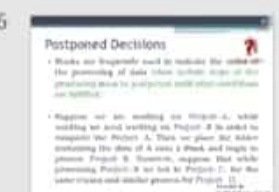
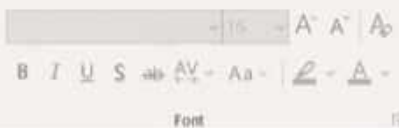
Postponed Decisions



- Stacks are frequently used to indicate the order of the processing of data when certain steps of the processing must be postponed until other conditions are fulfilled.
- Suppose we are working on Project-A, while working we need working on Project-B in order to complete the Project-A. Then we place the folder containing the data of A onto a Stack and begin to process Project-B. However, suppose that while processing Project-B we led to Project-C, for the same reason and similar process for Project-D.

Presented By:
Sheikh Muhammad Aamir

File Home Insert Design Transitions Animations Slide Show Review View Help Foxit Reader PDF



Postponed Decisions...



Study Point

- On the other hand, suppose we are able to complete the processing of **D**. Then the only project we may continue to process is **Project-C**, which is on the top of the Stack. Hence, we remove **C** from the stack and continue to process **C**.
- Similarly, after completing the processing of **C**, we remove **B** from the stack and continue the process **B**. Finally, after completing the processing of **B**, we remove the last folder of **A** from the stack, leaving the Stack empty and continue the processing of our original **Project-A**.

Presented By:
Sheikh Muhammad Aamir

Click to add notes

Paste New Slide Reuse Slides Layout Reset Section

Font B I U S A A A

Paragraph Text Direction Align Text Convert to SmartArt

Drawing Shape Fill Shape Outline Shape Effects Arrange Quick Styles

Editing Find Replace Select

Share Comments Dictate Design Ideas Voice Designer

5 LIFO Last In First Out (LIFO) data structure
6 Postponed Decisions
7 Postponed Decisions...
8 Postponed Decisions...
9 Postponed Decision...

Postponed Decisions...


Study Point

		C			
	B	B	B		
A	A	A	A	A	

Presented By:
Sheikh Muhammad Aamir

Click to add notes

- 7 Postponed Decisions...
- 8 Postponed Decisions...
- 9 Postponed Decision...
- 10 Operations on Stack
- 11 Push Method



Postponed Decision...

$$n! = n(n-1)!$$

$$(n-1)! = (n-1)(n-2)!$$

$$4! = 4(4-1)! = 4 * 3!$$

$$3! = 3(3-1)! = 3 * 2!$$

$$2! = 2(2-1)! = 2 * 1!$$

$$1! = 1$$

$$2! = 2 * 1 = 2$$

$$3! = 3 * 2 = 6$$

$$4! = 4 * 6 = 24$$

Presented By:
Sheikh Muhammad Aamir

Click to add notes

7 Postponed Decisions...

8 Postponed Decisions...

9 Postponed Decision...

10 Operations on Stack

11 Push Method

10

Operations on Stack

- Push operation for Insertion
- Pop operation for Deletion

Study Point

Presented By:
Sheikh Muhammad Aamir

9 Postponed Decision...

10 Operations on Stack


11 Push Method

12 Pop Method

13 Array Representation of Stacks

11

Push Method

 Study Point

- push() method adds an element at the top the stack.
- It takes as argument an Object to be pushed.
- It first checks if there is room left in the stack. If no room is left, it throws an exception. Otherwise, it puts the object into the stack array, by incrementing TOP variable by one.

Presented By:
Sheikh Muhammad Aamir

AutoSave Off DSA Stack and its application - Saved to this PC Search aamir

File Home Insert Design Transitions Animations Slide Show Review View Help Foxit Reader PDF

Paste New Slide Reuse Slides Layout Reset Section

Clipboard Slides

Font Paragraph Drawing Editing Voice Designer

9 Postponed Decision...

10 Operations on Stack

11 Push Method

12 Pop Method


13 Array Representation of Stacks

Click to add notes

Slide 12 of 51

12

Pop Method


Study Point

- The **pop method** removes an item from the stack and returns that item.
- The **pop method** first checks if the stack is empty. If the stack is empty, it throws an exception like Stack is empty. Otherwise, it simply **decreases TOP by one and returns the item found at the top of the stack.**

Presented By:
Sheikh Muhammad Aamir


Array Representation of Stacks Study Point

- Stack is maintained by a linear array, an indicator **TOP**, which contains the location of **top element** of the stack; and a variable N which represent the maximum size of array STACK.
- The **TOP = 0** or **TOP = NULL** will indicate that the stack is empty.

Presented By:
Sheikh Muhammad Aamir

- 11 Push Method
- 12 Pop Method
- 13 Array Representation of Stacks
- 14 Another Example of Stack
- 15 Stack operations using array...

14



Another Example of Stack

Stack with size $N = 7$
TOP = 0 (indicates the Stack is empty)

1	2	3	4	5	6	7

PUSH(25)
TOP

1	2	3	4	5	6	7
25						

PUSH(10)
PUSH(50)
PUSH(75)


1	2	3	TOP	5	6	7
25	10	50	75			

Click to add notes

- 13 Array Representation of Stacks
- 14 Another Example of Stack
- 15 Stack operations using array...
- 16 Push Procedure
- 17 Pop Procedure

15

Stack operations using array...



POP() [returns 75 from TOP]

TOP

1	2	3	4	5	6	7
25	10	50				

POP() [returns 50 from TOP]

TOP

1	2	3	4	5	6	7
25	10					

PUSH(55) [insert 55 into the stack]

TOP

1	2	3	4	5	6	7
25	10	55				

Presented By:
Sheikh Muhammad Aamir

Click to add notes

AutoSave Off DSA Stack and its application - Saved to this PC Search aamir

File Home Insert Design Transitions Animations Slide Show Review View Help Foxit Reader PDF

Clipboard Paste New Slide Reuse Slides Layout Reset Section Slides

Font Paragraph Drawing Editing Voice Designer

13 Array Representation of Stacks

14 Another Example of Stack

15 Stack operations using array...

16 Push Procedure

17 Pop Procedure

16

Push Procedure

Push (STACK, TOP, X, N)

1. If $TOP \geq N$ Then [Is stack already filled ?]
Write : Stack is Full
Else:
Set $TOP := TOP + 1$ [Increasing TOP by 1]
Set $STACK[TOP] := X$ [Insert X item at new TOP position]
[End of If structure]
2. Return

Study Point


Presented By:
Sheikh Muhammad Aamir

Click to add notes

Slide 16 of 51

- 15 Stack operations using array...
- 16 Push Procedure
- 17 Pop Procedure
- 18 Practice
- 19 Application of Stacks: Postfix Expression Calculator

17



Study Point

Pop Procedure

POP (STACK, TOP, N)

1. If TOP = 0 Then

[Is stack already empty ?]

Write: Stack Empty

Return
- Else:

Set X := STACK[TOP] [Assigns TOP element to X]

Set TOP := TOP - 1 [TOP decreases by 1]

Return X

[end of if structure]

Presented By:

Sheikh Muhammad Aamir

Practice



- Implement the Stack Operations Push and Pop in C++.
- Construct an algorithm for calculating to convert a decimal number into a binary number using Push and Pop of Stack functions.
- Write a program in C ++ using Stack to convert a Decimal number to Binary number.

Presented By:
Sheikh Muhammad Aamir

Applications of Stack: Arithmetic Expressions Solving

- Let Q be an expression that contains
 - operands and operators
- Expression can be written in three format
 - Infix Notation
 - Prefix Notation
 - Postfix Notation
- Evaluation of Q by using reverse Polish (Postfix) notation
- Prefix Expression is known as Polish Notation
- Suffix/Postfix Expression is known as Reverse Polish Notation

Presented By:
Sheikh Muhammad Aamir

Arrange

Arrange objects on the slide by changing their order, position, and rotation.

You can also group multiple objects together so that they'll be treated like a single object.

suuuy rouu

Click to add notes

19 Applications of Stack:
Arithmetic Expression Solving

- Right to left expression evaluation
- Expression with infix notation (Polish Notation)
- Infix notation
- Infix notation with binary operators (Arithmetic)
- Infix notation with unary operators (Arithmetic)
- Infix notation with parentheses (Arithmetic)
- Infix notation with operators and parentheses (Arithmetic)

20 Expressions

- Infix Notation e.g. $A + B$
- Prefix Notation (Polish Notation) e.g. $+ AB$
- Postfix Notation (Reverse Polish Notation) e.g. $AB +$

21 Operator Precedence

- Highest: Exponentiation ($^$), $\sqrt{\quad}$
- Next Highest: Multiplication ($*$) and Division ($/$)
- Lowest: Addition ($+$) and Subtraction ($-$)
- All other operators have lower precedence

22 Expression Evaluation: Example

```

1 1 + 2 * 3 - 4 * 5
2 1 + 6 - 20
3 7 - 20
4 -13
5 -13

```

23 Prefix / Polish and Reverse Polish Notation: Arithmetic Expression Conversion

20

Expressions

- Infix Notation
e.g. $A + B$
- Prefix Notation (Polish Notation)
e.g. $+ AB$
- Postfix Notation (Reverse Polish Notation)
e.g. $AB +$

Here A, and B are operands whereas + is an operator

Presented By:
Sheikh Muhammad Aamir

19 Applications of Stack: Arithmetic Expression Solving

20 Expressions

21 Operator Precedence

22 Expression Evaluation: Example

23 Prefix, Infix and Reverse Polish Notation: Mutual Conversion

21

Operator Precedence

- Highest: Exponentiation ($^$, $\$$, \uparrow)
- Next Highest: Multiplication ($*$) and Division ($/$)
- Lowest: Addition ($+$) and Subtraction ($-$)
- All these precedence for binary operations

Presented By:
Sheikh Muhammad Aamir

19 Applications of Stack: Arithmetic Expression Solving


20 Expressions

21 Operator Precedence

22 Expression Evaluation: Example

23 Prefix, Infix and Reverse Polish Notation: Recursive Computation

Expression Evaluation: Example



$$\begin{aligned} &2^3 + 5 * 2^2 - 12 / 6 \\ &= 8 + 5 * 4 - 12 / 6 \\ &= 8 + 20 - 2 \\ &= 28 - 2 \\ &= 26 \end{aligned}$$

Presented By:
Sheikh Muhammad Aamir

21

Operator Precedence

- High: Exponentiation (^), & (%)
- Next Highest: Multiplication (*) & Division (/)
- Lowest: Addition (+) & Subtraction (-)

22

Expression Evaluation: Example

$$A = 7 + 8 * 9 - 10 / 4$$

$$= 7 + 72 - 2.5$$

$$= 76.5$$

23

Prefix / Polish and Reverse Polish (Postfix) Notation: Manual Conversion

A + B	(A + B) * C	A + (B * C)	(A + B) / (C - D)
$= +AB$	$= +AB * C$ $= * +ABC$	$= A + *BC$ $= + A * BC$	$= +AB / -CD$ $= / +AB - CD$

Prefix Examples

A + B	(A + B) * C	A + (B * C)	(A + B) / (C - D)
$= AB +$	$= AB + * C$ $= AB + C *$	$= A + BC *$ $= ABC * +$	$= AB + / CD -$ $= AB + CD - /$

Postfix Examples

24

Manual Conversion for Prefix and Postfix Conversion

Prefix Conversion:

Expression	Prefix
(A + B) * C	* + AB C
A + (B * C)	+ A * BC
(A + B) / (C - D)	+ / AB - CD

Postfix Conversion:


Expression	Postfix
(A + B) * C	AB + C *
A + (B * C)	ABC * +
(A + B) / (C - D)	AB + CD - /

25

Click to add notes

23

Prefix / Polish and Reverse Polish (Postfix) Notation: Manual Conversion




Study Point

Presented By:
Sheikh Muhammad Aamir

Slide 23 of 50

Scanned with CamScanner

24



Study Point

Manual Infix to Prefix and Postfix Conversion ...

Infix Expression	Prefix Expression	Postfix Expression
$(A+B) * (C-D)$	$= +AB * - CD$ $= *+AB - CD$	$= AB+ * CD-$ $= AB+ CD- *$
$A + B * C - (D * 2 * 3)$	$A + B * C - (\$ D2 * 3)$ $= A + B * C - (* \$ D2 3)$ $= A + * B C - (* \$ D2 3)$ $= + A * B C - * \$ D2 3$ $= - + A * B C * \$ D2 3$	$A + B * C - (D2 \$ * 3)$ $= A + B * C - (D2 \$ * 3)$ $= A + B C * - D2 \$ 3 *$ $= A B C * + - D2 \$ 3 *$ $= A B C * + D2 \$ 3 * -$

Presented By:
Sheikh Muhammad Aamir

AutoSave Off DSA Stack and its application Search aamir

File Home Insert Design Transitions Animations Slide Show Review View Help Foxit Reader PDF

Paste New Slide Reuse Slides Layout Reset Section

Clipboard Slides

Font Paragraph Drawing Editing Voice Designer

Find Replace Select Dictate Design Ideas

23 Prefix, Infix and Reverse Polish (Postfix) Notation: Manual Conversion

24 Reverse Infix to Prefix and Postfix Conversion

25 For Practice
Convert Infix to Prefix and Postfix Notation Manually

Study Point

1. $A + B * (C + D) / F + D * E$
2. $(A + B) / C * D - E$
3. $12 / (7 - 3) + 2 * (1 + 5)$

26 Expression Solving using STACK

27 Extension of a Prefix Expression using Stack

Click to add notes

Slide 25 of 30

Presented By:
Sheikh Muhammad Aamir

25

26

27


28

29

Click to add notes

26

Expression Solving using STACK

 Study Point

1. Infix to Postfix Conversion using STACK
2. Evaluation of Postfix Expression using STACK

Presented By:
Sheikh Muhammad Aamir

25



26



27



28



29



1. Transforming Infix Expression into Postfix Expression using Stack



Study Point

POLISH(Q, P)
Suppose Q is an arithmetic expression written in infix notation. This algorithm finds the equivalent postfix expression P.

1. Push "(" onto STACK, and add ")" to the end of Q.
2. Scan Q from left to right and repeat Steps 3 to 6 for each element of Q until the STACK is empty:
 - 3. If an operand is encountered, add it to P.
 - 4. If a left parenthesis is encountered, push it onto STACK.
 - 5. If an operator \odot is encountered, then:
 - (a) Repeatedly pop from STACK and add to P each operator (on the top of STACK) which has the same precedence as or higher precedence than \odot .
 - (b) Add \odot to STACK.
 - 6. If a right parenthesis is encountered, then:
 - (a) Repeatedly pop from STACK and add to P each operator (on the top of STACK) until a left parenthesis is encountered.
 - (b) Remove the left parenthesis. [Do not add the left parenthesis to P.]

[End of If structure.]
[End of Step 2 loop.]
Exit

Prepared By:
Sheikh Muhammad Aamir

Click to add notes

37

38

39

40

41

40

Transforming Infix Expression into Postfix Expression using Stack : Example

- Infix Expression will be as : $A * (B + C) - D / E$

Now reading last symbol

Study Point

Sr.	Scanned Symbol	STACK	Expression P
1		(
2	A	(A
3	*	(*	A
4	((*(A
5	B	(*(AB
6	+	(*(+	AB
7	C	(*(+	ABC
8)	(*	ABC+
9	-	(-	ABC+*
10	D	(-	ABC+*D
11	/	(-/	ABC+*D
12	E	(-/	ABC+*DE
13)		ABC+*DE/-

39

40


41

42

43

41

Transforming Infix Expression into Postfix Expression using Stack: Practice Questions


Study Point

1. $A / B - C + D * E - A * C$
2. $(A + B) / C * D - E$
3. $A + B * (C ^ D - E)$
4. $(A - B) * (A + B)$
5. $2 * 3 + 5 * 4 - 9$

I

Presented By:
Sheikh Muhammad Aamir

42 Expression Solving using STACK

43 Evaluation of a Prefix Expression using Stack


44 Evaluation of Prefix using Stack: Example

45 Evaluation of Prefix using Stack: Example

46 Evaluation of Prefix using Stack: Example

42

Expression Solving using STACK

 Study Point

1. Infix to Postfix Conversion using STACK
2. Evaluation of Postfix Expression using STACK

Presented By:
Sheikh Muhammad Aamir

2. Evaluation of a Postfix Expression using Stack

Study Point

This algorithm finds the VALUE of an arithmetic expression P written in postfix notation.


1. Add a right parenthesis ")" at the end of P [This acts as a sentinel]
2. Scan P from left to right and repeat Steps 3 and 4 for each element of P until the sentinel ")" is encountered.
3. If an operand is encountered, put it on STACK.
4. If an operator \odot is encountered, then:
 - (a) Remove the two top elements of STACK, where A is the top element and B is the next-to-top element.
 - (b) Evaluate $B \odot A$.
 - (c) Place the result of (b) back on STACK.
 [End of If structure.]
5. Set VALUE equal to the top element on STACK.
6. Exit

Presented By:
Sheikh Muhammad Aamir

- 42 Expression Solving using STACK
- 43 Evaluation of a Postfix Expression using Stack
- 44 Evaluation of Postfix using Stack: Example
- 45 Evaluation of Postfix using Stack: Example
- 46 Evaluation of Postfix using Stack: Example

44

Evaluation of Postfix using Stack: Example

 Study Point

Infix Expression (Q): $5*(6+2)-(12/4)$

Postfix Expression (P): $5, 6, 2, +, *, 12, 4, /, -$

Step 1: Place) at the end of P

Postfix Expression (P): $5, 6, 2, +, *, 12, 4, /, -,)$


Presented By:
Sheikh Muhammad Aamir

- 52 Evaluation of Postfix using Stack: Example
- 53 Evaluation of Postfix using Stack: Example
- 54 Evaluation of Postfix using Stack: Example
- 55 Evaluation of postfix expression using Stack: Practice Questions
- 56

54

Evaluation of Postfix using Stack: Example

Postfix Expression: 5, 6, 2, +, *, 12, 4, /, -,)

 Study Point

Sr.	Scanned Symbol	STACK
1	5	5
2	6	5, 6
3	2	5, 6, 2
4	+	5, 8
5	*	40
6	12	40, 12
7	4	40, 12, 4
8	/	40, 3
9	-	37
10)	

Presented By:
Sheikh Muhammad Aamir

52 Evaluation of Postfix using Stack: Example

53 Evaluation of Postfix using Stack: Example


54 Evaluation of Postfix using Stack: Example

55 Evaluation of Postfix Expression using Stack: Practice Questions

56

55

Evaluation of Postfix Expression using Stack: Practice Questions

 Study Point

1. 2, 3, 1, *, +, 9, -
2. 7, 2, 3, *, 5, +, 8, 4, 2, /, -, *, -
3. A, B, C, D, 2, ^, *, +, 3, +, +

Presented By:
Sheikh Muhammad Aamir