Example:

Algorithm: (Largest Element in Array) Given a non-empty array DATA with N values, this algorithm finds the location LOC and value MAX of the largest element of DATA

Study Point

1. [Initialize] Set K :=1, LOC := 1, and MAX :=DATA[1]
2. Repeat Step 3 and step 4 while K ≤ N:
3. If MAX < DATA [K], Then:
   Set MAX := DATA[K] and LOC := K
   [End of If structure]
4. Set K := K + 1
   [End of Step 2 loop]
5. Write LOC, MAX
6. Exit

Presented By:
Sheikh Muhammad Aamir

Algorithm: (Quadratic Equation) This algorithm input the coefficients A, B, C of quadratic equation and output the real solutions, if any.

Study Point

Step 1 Read: A, B, C.
Step 2 Set D := $B^2 - 4AC$.
Step 3 If D > 0, Then
(a) Set X1 := (-B + √D) / 2A
Set X2 := (-B - √D) / 2A
(b) Write : X1, X2
Else If D = 0, Then
(a) Set X1 := (-B) / 2A
(b) Write : X1
Else:
Write : 'No real value solution'
[End of If structure]
Step 4 Exit.

Presented By:
Sheikh Muhammad Aamir

23

Iteration Logic

Algorithm:

(Linear Search) A linear array DATA with N elements and a specific ITEM of information are given. This algorithm finds the location LOC of ITEM in the array DATA or sets LOC = 0.

1. [Initialize] Set K := 1 and LOC := 0.
2. Repeat Steps 3 and 4 while LOC = 0 and K <= N.
3. If ITEM = DATA[K], then: Set LOC: =K.
4. Set K := K + 1. [Increments counter.]
   [End of Step 2 loop.]

1	2	3	4	5
10	25	15	22	17

K = 1, N=5 LOC = 0

ITEM = 15 (to be searched)

5. [Successful?]
   If LOC = 0, then:
   Write: ITEM is not in the array DATA.
   Else:
   Write: LOC is the location of ITEM.
   [End of If structure.]
6. Exit.

Presented By:
Sheikh Muhammad Aamir

Algorithm:

(Linear Search) A linear array DATA with N elements and a specific ITEM of information are given. This algorithm finds the location LOC of repeated ITEM in the array DATA or sets LOC = 0.

Study Point

1. [Initialize] Set K := 1 and LOC := 0.
2. Repeat Steps 3 and 4 while ~~LOC = 0 and~~ K <= N.
3. If ITEM = DATA[K], then:
   i. Set LOC: =K
   ii. Write: "Element found at location", LOC
1. Set K := K + 1. [Increments counter.]
   [End of Step 2 loop.]

1	2	3	4	5
10	25	15	22	15

K = 1, N=5 LOC = 0

ITEM = 15 (to be searched)

5. [Successful?]
   If LOC = 0, then:
   Write: ITEM is not in the array DATA.
   ~~Else:~~
   ~~Write: LOC is the location of ITEM.~~
   [End of If structure.]
6. Exit.

Presented By:
Sheikh Muhammad Aamir

Write a procedure FIND(DATA, N, LOC1, LOC2) which finds the location LOC1 of the largest element and the location LOC2 of the second largest element in an array DATA with n > 1 elements.

Study Point

Logic:

The elements are examined one by one

Each new element DATA[K] is tested as follows; If

SECOND <= FIRST < DATA[K]

Then FIRST becomes the new SECOND and DATA[K] becomes the new FIRST element

On the other hand, if

SECOND < DATA[K] < FIRST

Then DATA [K] becomes the new SECOND element.

Initially, set FIRST = DATA[1] and SECOND = DATA[2] and check weather or not they are in the right order.

Presented By:
Sheikh Muhammad Aamir

Write a procedure FIND(DATA, N, LOC1, LOC2) which finds the location LOC1 of the largest element and the location LOC2 of the second largest element in an array DATA with n > 1 elements.

Study Point

FIND(DATA, N, LOC1, LOC2)

1. Set FIRST := DATA[1], SECOND := DATA[2], LOC1 := 1, LOC2 := 2.
2. [Are FIRST and SECOND initially correct?]
   If FIRST < SECOND, then:
   (a) Interchange FIRST and SECOND.
   (b) Set LOC1 := 2 and LOC2 := 1.

   [End of If structure.]
3. Repeat for K = 3 to N:
   If FIRST < DATA[K], then:
   (a) Set SECOND := FIRST and FIRST := DATA[K].
   (b) Set LOC2 := LOC1 and LOC1 := K.
   Else if SECOND < DATA[K], then:
   Set SECOND := DATA[K] and LOC2 := K.
   [End of If structure.]
   [End of loop.]
4. Return.

Presented By:
Sheikh Muhammad Aamir

1	2	3	4	5	6
15	75	55	85	35	45

Study Point

Initially suppose

FIRST = DATA[1], SECOND = DATA[2] i.e.

FIRST = 15, SECOND = 75

LOC1 = 1, LOC2 = 2

Initial order checking

If FIRST < SECOND then

Swap FIRST, SECOND and LOC1 = 2, LOC2 = 1

Here the case is true, so FIRST = 75, SECOND = 15

Loop iterations K= 3 to 6

If FIRST < DATA[K] then

Set SECOND = FIRST, and FIRST = DATA[K] and LOC2 = LOC1, LOC1 = K

Else if SECOND < DATA[K] then

Set SECOND = DATA[K] and LOC2 = K

Here for K= 3, FIRST = 75, SECOND = 55, LOC1 = 2, LOC2 = 3

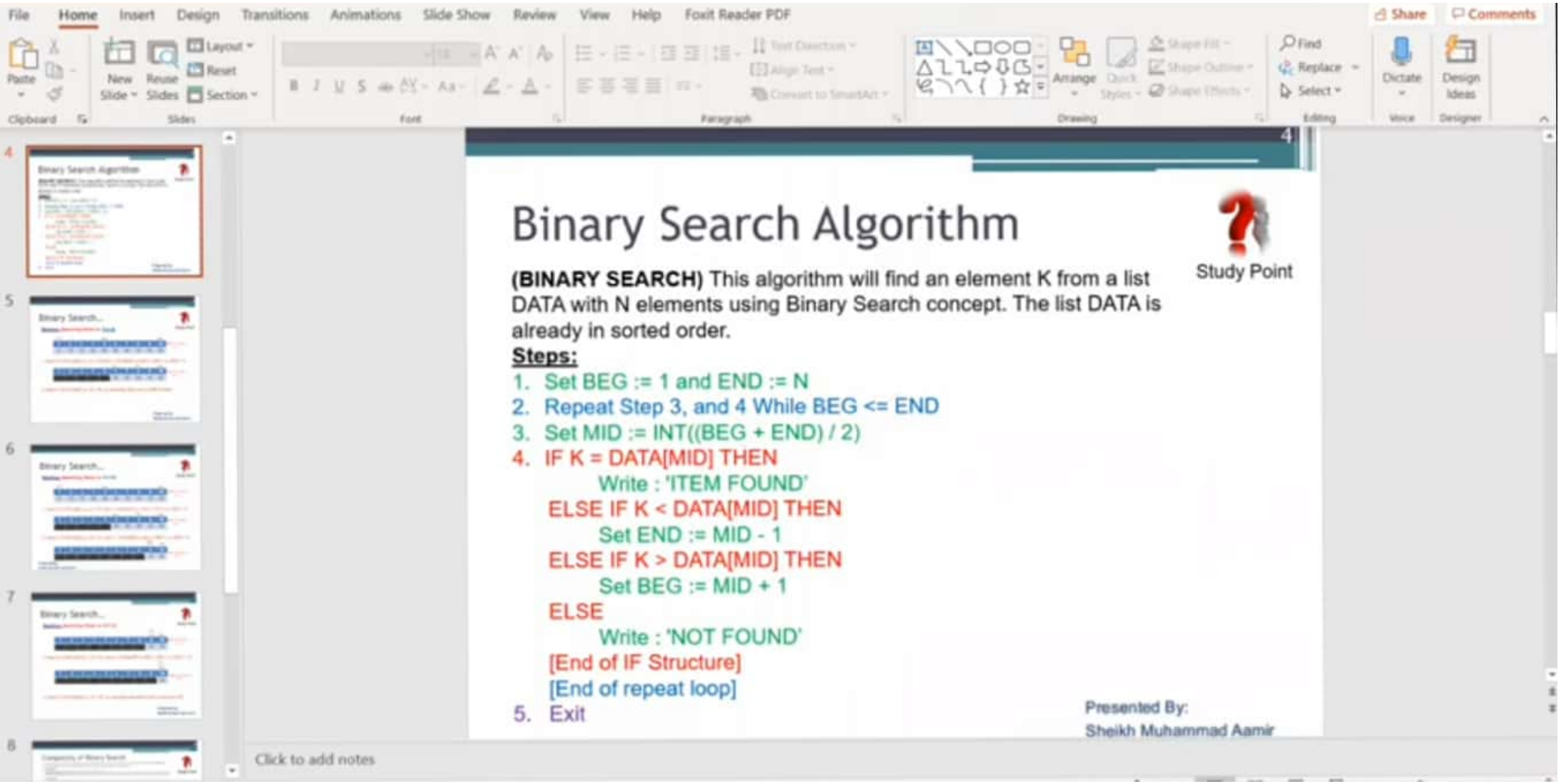
For K = 4, FIRST = 85, SECOND = 75, LOC2 = 2, LOC1 = 4

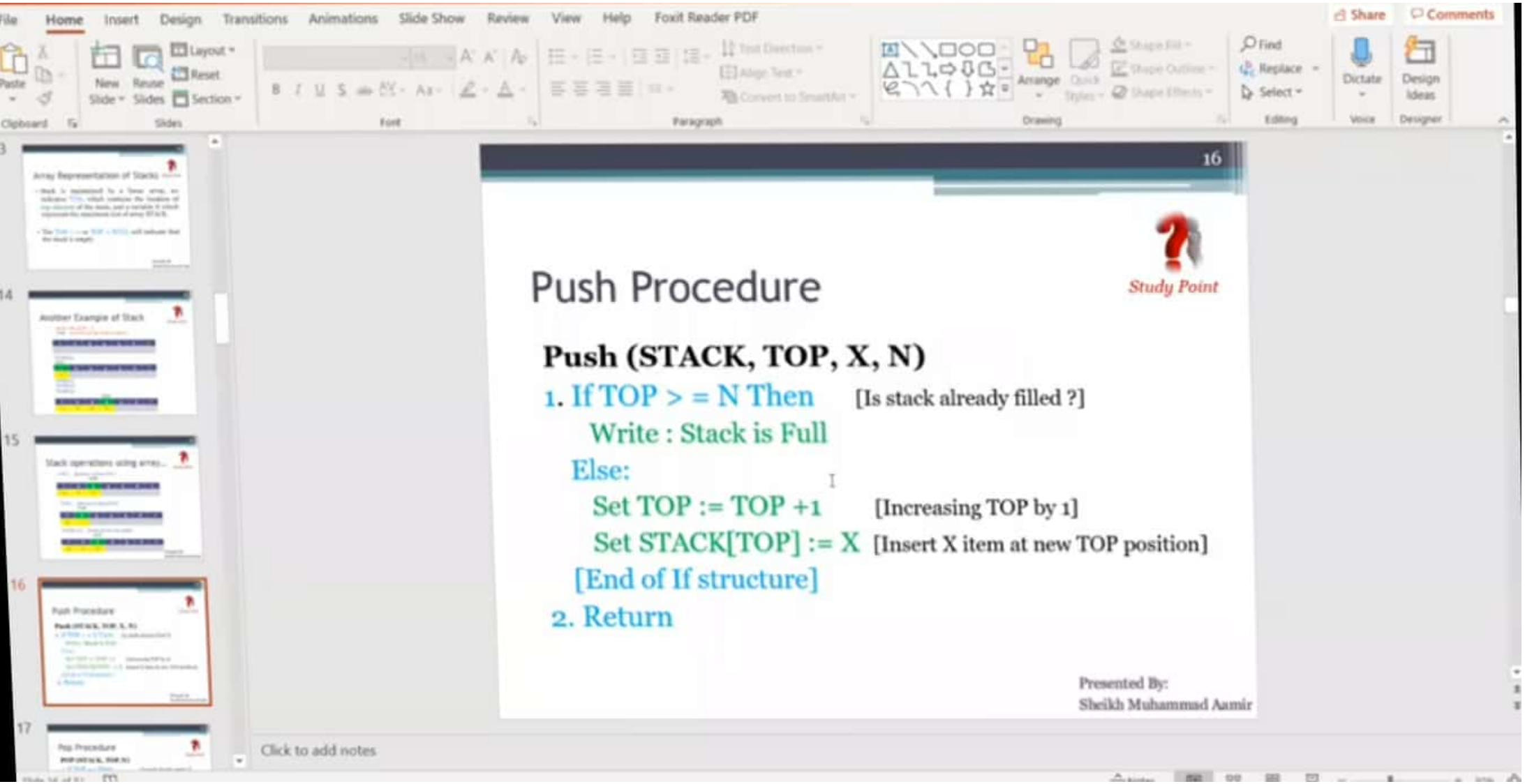
For K = 5, FIRST = 85, SECOND = 75, LOC2 = 2, LOC1 = 4 (NO CHANGE)

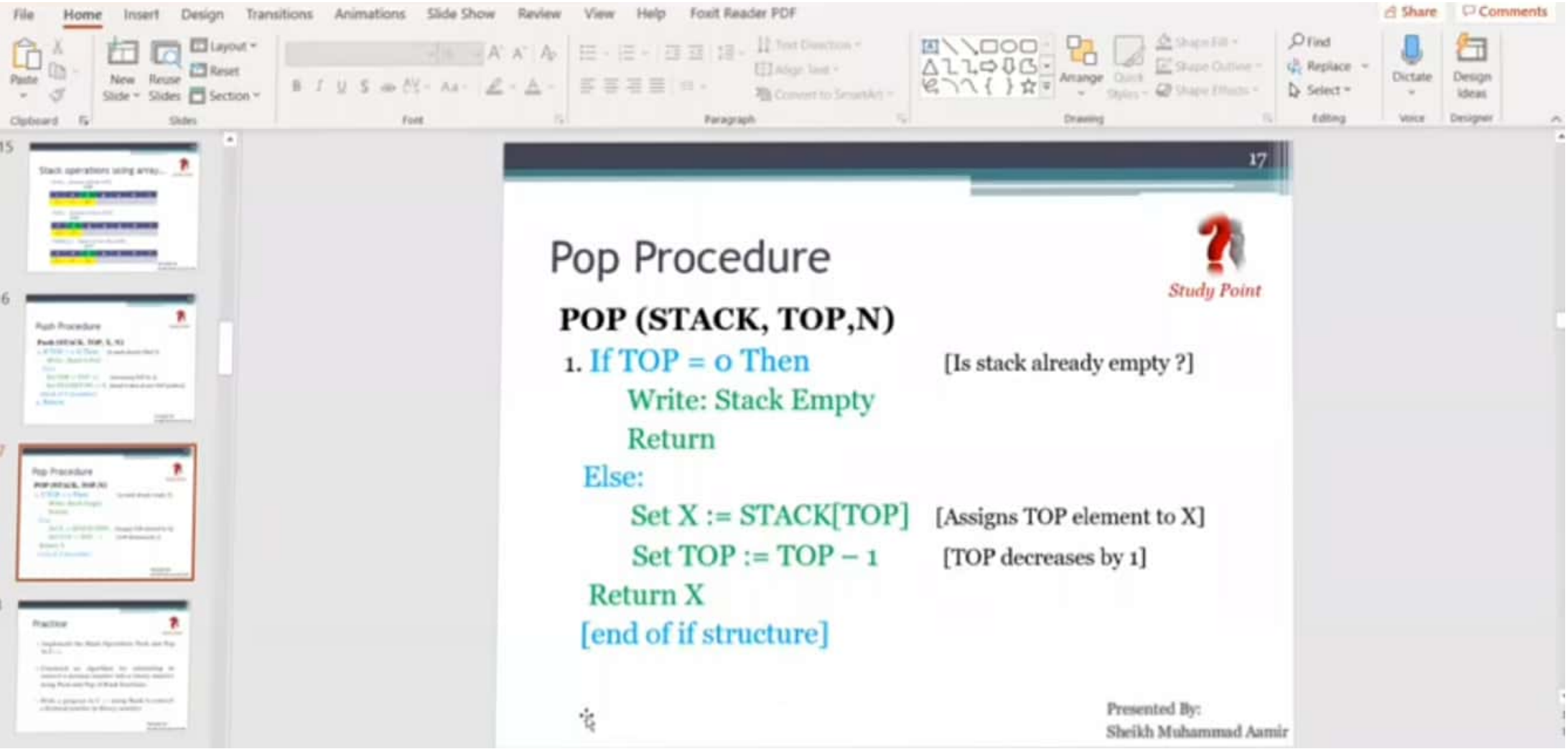
For K = 6, FIRST = 85, SECOND = 75, LOC2 = 2, LOC1 = 4 (NO CHANGE)

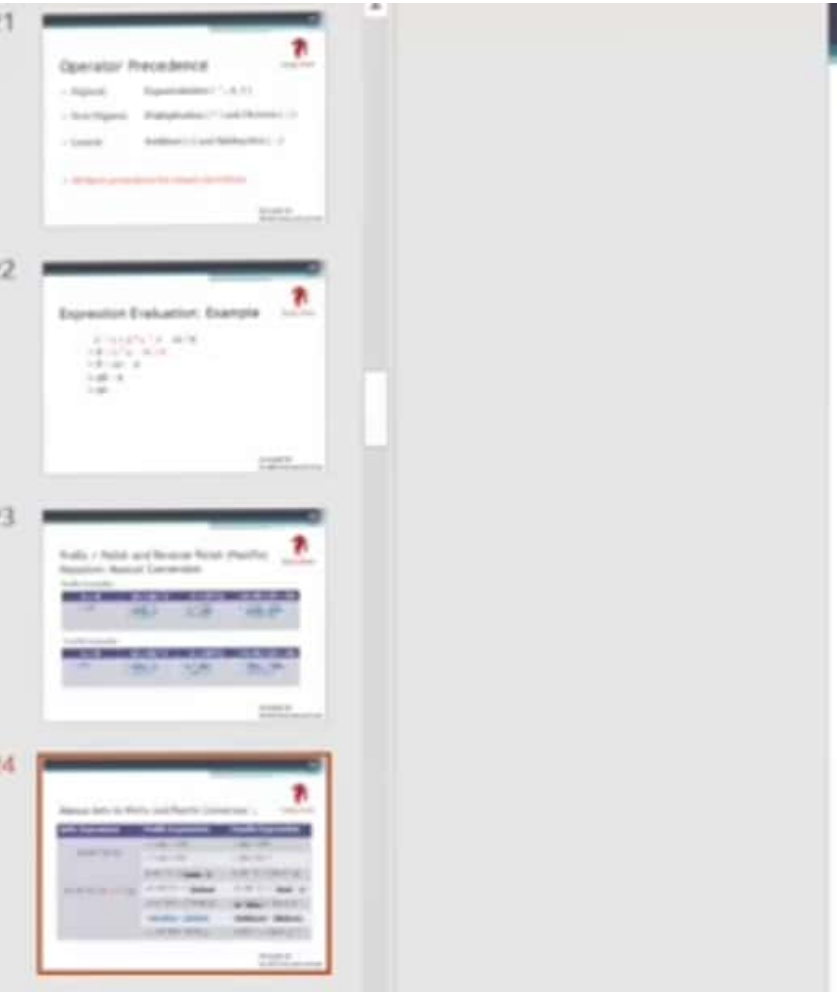
Presented By:

Sheikh Muhammad Aamir









Manual Infix to Prefix and Postfix Conversion ...

Study Point

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

Transforming Infix Expression into Postfix Expression using Stack : Example

Study Point

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

Now reading last symbol

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/-

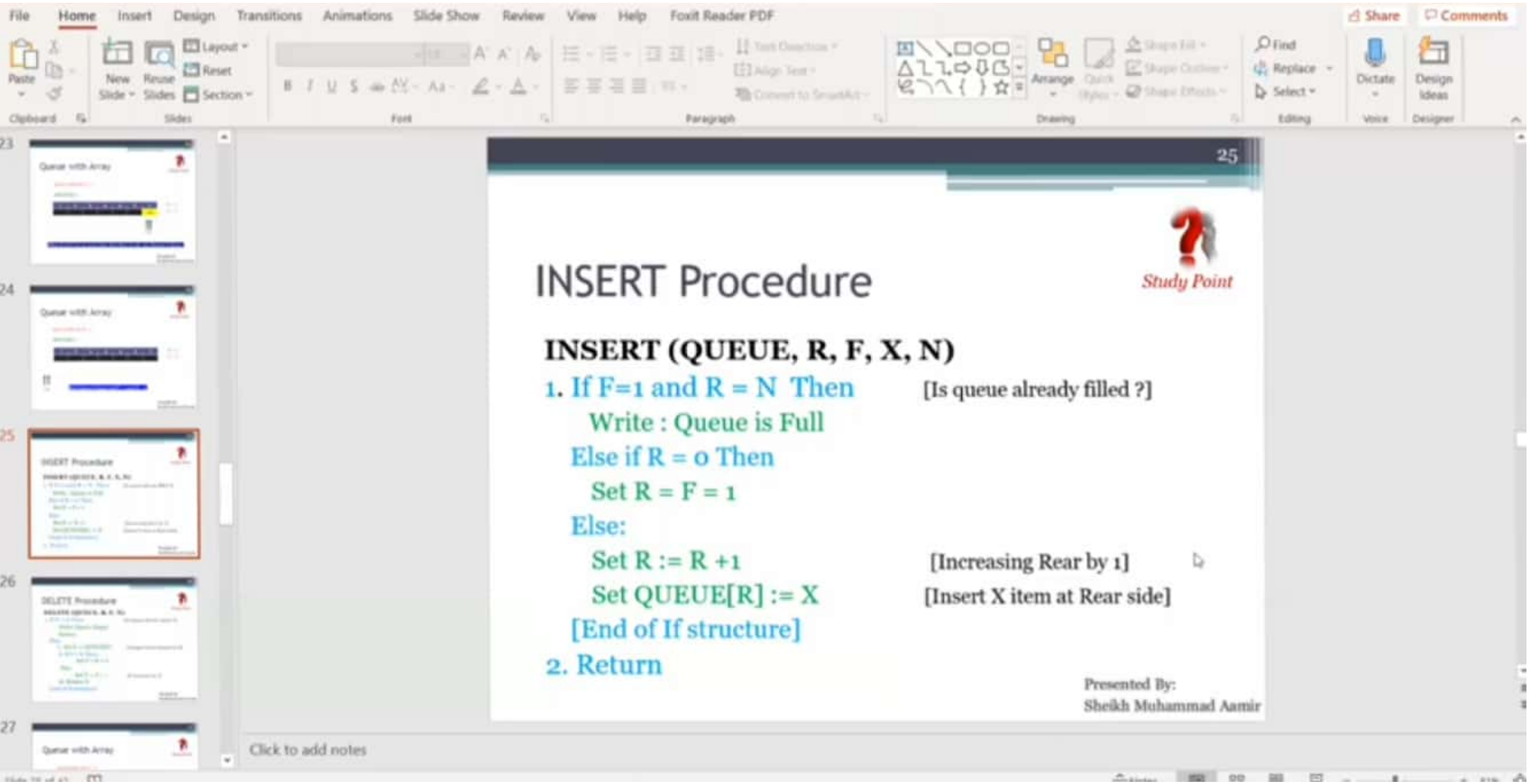
Evaluation of Postfix using Stack: Example

Study Point

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

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



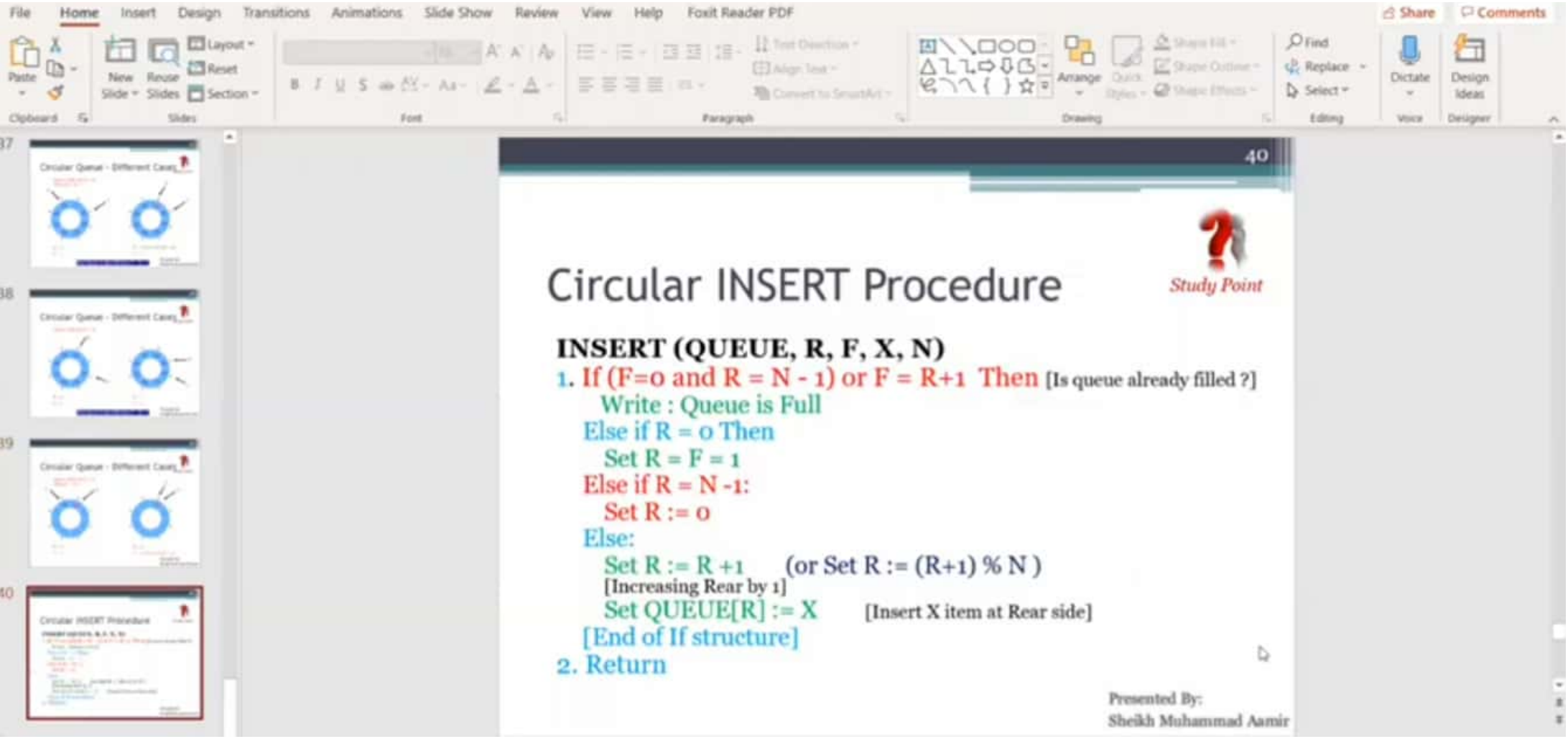
DELETE Procedure

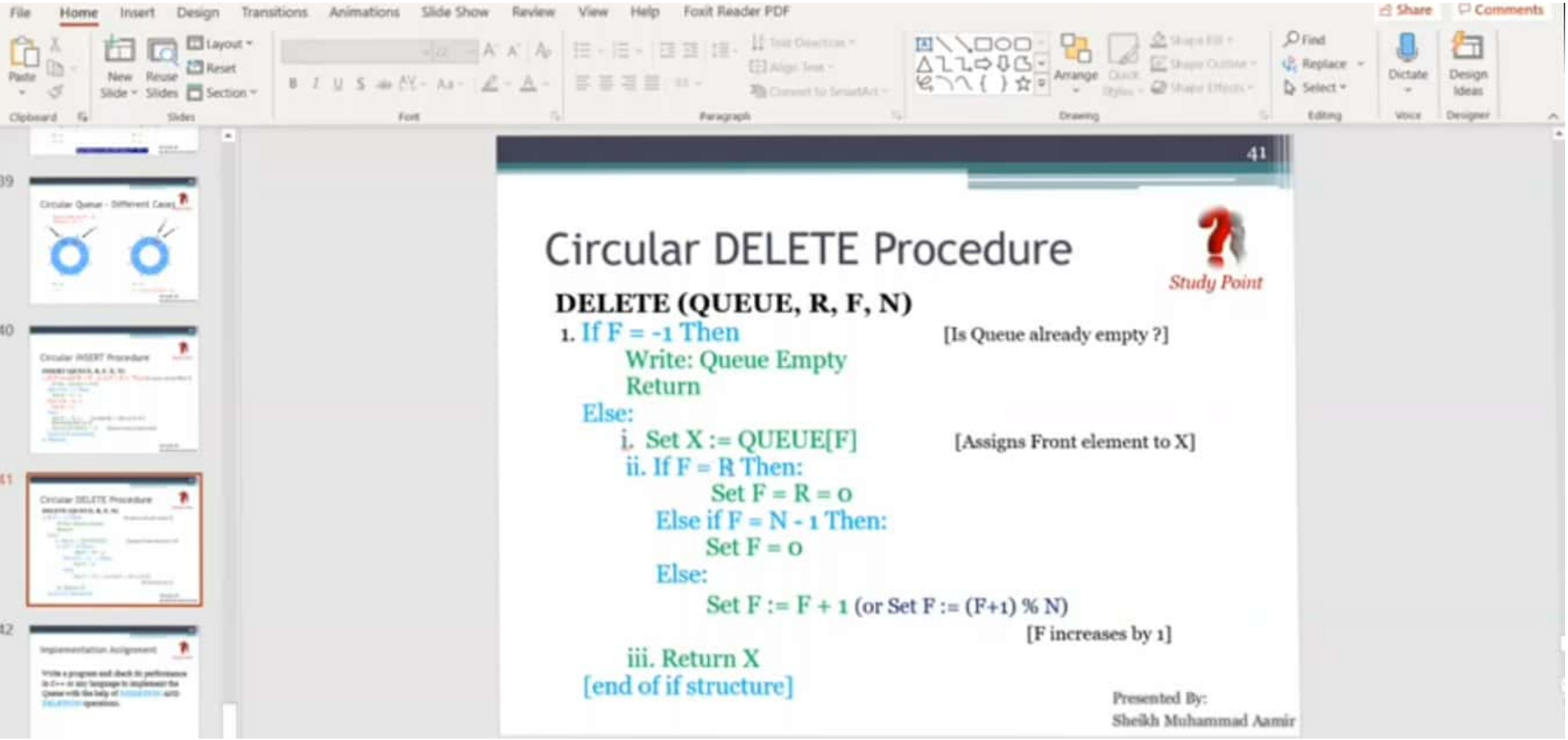
Study Point

DELETE (QUEUE, R, F, N)

1. If F = 0 Then [Is Queue already empty ?]
   Write: Queue Empty
   Return
   Else:
   i. Set X := QUEUE[F] [Assigns Front element to X]
   ii. If F = R Then:
   Set F = R = 0
   Else:
   Set F := F + 1 [F increases by 1]
   iii. Return X
   [end of if structure]

Presented By:
Sheikh Muhammad Aamir



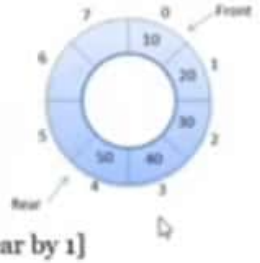


Insertion from Rear side

Study Point

INSERT_REAR (QUEUE, R, F, X, N)

1. If (F=0 and R = N - 1) or F = R+1 Then [Is queue already filled ?]
   Write : Queue is Full
   Else if R = -1 Then [first time insertion]
   Set R = F = 0
   Else if R = N -1:
   Set R := 0
   Else:
   Set R := R +1 [Increasing Rear by 1]
   Set QUEUE[R] := X [Insert X item at Rear side]
   [End of If structure]
2. Return



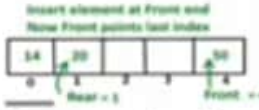
Presented By:
Sheikh Muhammad Aamir

Insertion from Front side

Study Point

INSERT_FRONT (QUEUE, R, F, X, N)

1. If (R=0 and F = N - 1) or R = F+1 Then [Is queue already filled ?]
   Write : Queue is Full
   Else if F = -1 Then [first time insertion]
   Set F=R=0
   Else if F = 0:
   Set F := N - 1
   Else:
   Set F := F - 1
   Set QUEUE[F] := X
   [End of If structure]
2. Return



[Decreases Front by 1]

[Insert X item at Front side]

Presented By:
Sheikh Muhammad Aamir

Deletion from Front side

Study Point

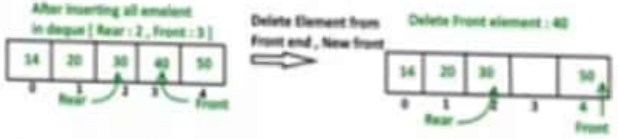
DELETE_FRONT (QUEUE, R, F, N)

1. If F = -1 Then
   Write: Queue Empty
   Return
   Else:
   i. Set X := QUEUE[F]
   ii. If F = R Then:
   Set F = R = -1
   Else if F = N - 1 Then:
   Set F = 0
   Else:
   Set F := F + 1
   iii. Return X
   [end of if structure]

[Is Queue already empty ?]

[Assigns Front element to X]
[only one element in queue]

[F increases by 1]



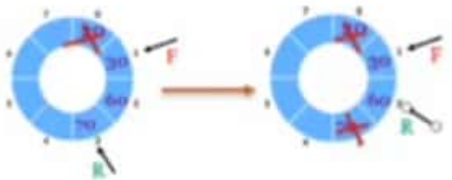
Click to add notes

Deletion from Rear side

Study Point

DELETE_REAR (QUEUE, R, F, N)

1. If R = -1 Then [Is Queue already empty ?]
   Write: Queue Empty, so cannot delete from Rear end
   Return
   Else:
   i. Set X := QUEUE[R] [Assigns Rear element to X]
   ii. If F = R Then:
   Set F = R = -1
   Else if R = 0 Then:
   Set R = N - 1
   Else:
   Set R := R - 1
   iii. Return X
   [end of if structure]



Presented By:
Sheikh Muhammad Aamir