DSA THEORY ASSIGNMENT #3

Name: Ali ur Rehman
Student ID: 67267
Signature: Ali

Question # 1 : Find the polynomial POLY 1 and POLY 2 stored in fig. (Linked list)

1. Beginning with POLY 1, Traverse the list by following the pointers to obtain the polynomial
$p_1(x) = 3x^5 - 4x^3 + 6x - 5$

2. Beginning with POLY 1, traverse the list by following the pointers to obtain the polynomial
$p_1(x) = 2x^8 + 7x^5 - 3x^2$.

	COEF	EXP	LINK
1	0	-1	5
2			
3	6	1	7
4	-3	2	10
5	3	5	8
6	2	8	9
7	-5	0	1
8	-4	3	3
9	7	5	4
10	0	-1	6

POLY 1 [1] POLY 2 [0]

Answer: Polynomial Linked List Traversal

A linked list representation of two polynomials using an array-based structure, with COEF, EXP, and LINK fields. Following the links starting from the specified indices to form complete polynomial expressions.

> POLY1 Traversal

Starts from Index 5

Index	COEF	EXP	LINK
5	3	5	8
8	-4	3	3
3	6	1	7
7	-5	0	1
1	0	-1	

Polynomial 1 $p_1(x)$:-

$$P_1(x) = 3x^5 - 4x^3 + 6x - 5$$

> POLY2 Traversal

Start from Index 6

Index	COEF	EXP	LINK
6	2	8	9
9	7	5	4
4	-3	2	10
10	0	-1	6

Polynomial 2 $p_2(x)$:

$$P_2(x) = 2x^8 + 7x^5 - 3x^2$$

Question#2 - AVL Tree insertion & deletion Scenerious. You are hired as a software engineer in a company that handles a dynamic database of real-time transaction. IDS. The requirement is to maintain balanced search performance as transactions are frequently inserted & deleted. To achieve this, the company has decided to implement an AVL Tree, a type of self-balancing Binary search Tree.

Problem Task:

Analysis:-

AVL Tree Insertion and Deletion.

Insertion and deletion nodes is an AVL Tree &

observe rotations & balance factors at each step.

→ Given Insertion Order:

{25, 20, 5, 34, 50, 30, 10, 15}

→ Insertion steps with Rotations & Tree states.

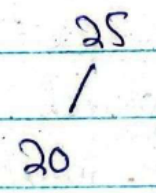
1. Insert 25

Tree = 25

Balance Factor = 0

Rotation = No rotations.

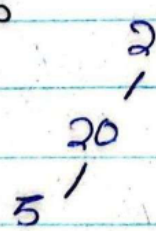
2. Insert 20



Balance Factor at 25 = +1

Rotation = No rotation.

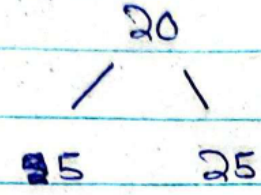
3. Insert 5



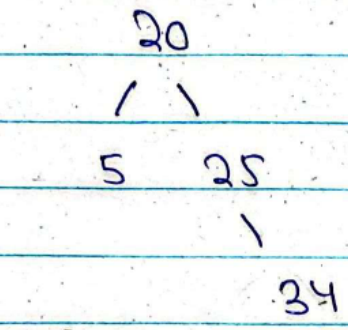
Balance Factor at 25 = +2 → LL Case

Rotation = Right Rotate at 25

After Rotation:



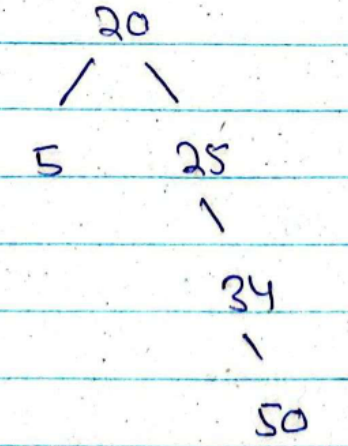
4. Insert 34



Balance Factor at 25 = -1

Rotation : No rotations.

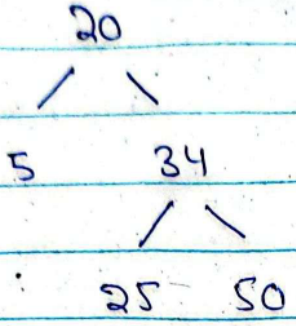
5. Insert 50



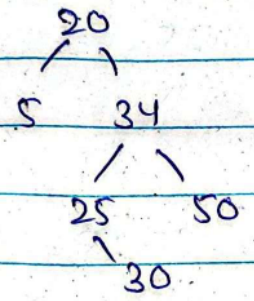
Balance factor at 25 = -2 → RR Case

Rotation = Left rotate at 25

After Rotation: (AVL Tree After First 5 Insertions)

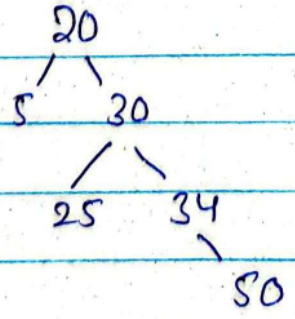


6. Insert 30

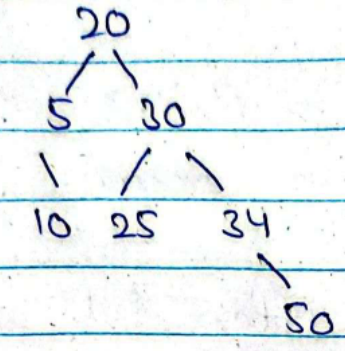


Imbalance at 34 -> RL case
Rotation = Right at 25, then left at 34

After rotation:

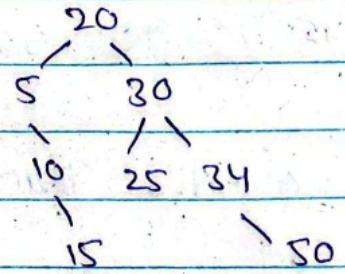


7. Insert 10



Rotation = No rotations

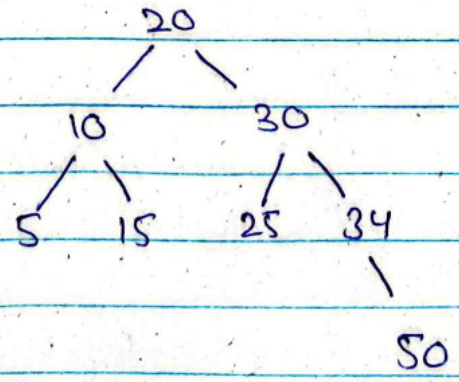
8. Insert 15



Imbalance at node 5 -> RR Case

Rotation = Left Rotate at 5

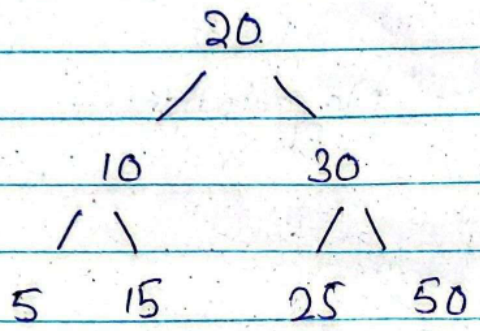
After Rotation : (AVL Tree After All insertions):



- Deletion of Node 34

Delete 34 → Replace with 50
No imbalance occurs.

AVL Tree After Deletion of 34:



Explanation of AVL Terminologies.

- **Insertion & Rotation:**

Inserting a node may unbalance the tree. AVL Trees fix it using rotations.

- **Single Rotation:**

- LL (Left-Left): Heavy on left child's left → Right Rotation

Example: Insert 5 → Rotation at 25

RR (Right - Right): Heavy on right child's right -> Left Rotation

Example: Insert 50 -> Rotation at 25

- **Double Rotation:**
  - LR (Left - Right): Left child is right-heavy -> Left the right Rotation.
  - RL (Right - Left): Right child is left-heavy -> Right then left Rotation

Example: Insert 30 -> RL rotation at 34

- **Outside Cases**

LL: Inserted in left of left -> Right Rotate
RR: Inserted in right of right -> Left Rotate

Inside Cases

LR: Inserted in right of left -> Double Rotate
RL: Inserted in left of right -> Double Rotate

Deletion:

Deleting a node in an AVL Tree may create imbalance in the height of subtrees. To restore balance, appropriate rotations (LL, RR, LR, RL) are applied.

In our case, deleting node 84 did not unbalance the tree, so no rotation was required.