**1. Program Analysis: Binary Search Tree**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| INSERT | Create a new Node and fill it  Find Correct PARENT  Place it  **Twist : If Root is NULL** | | | | |
| ***OR*** *Use the Typical Recursive Insert logic, Viz.* | | | | |
| DISPLAY | **Twist : If Root is NULL**  Use Recursion (in as follows) | | | | |
| For In-Order | | For Pre-Order | | For Post-Order |
| f(Root->Left)  Show Root->Element  f(Root->Right) | | Show Root->Element  f(Root->Left)  f(Root->Right) | | f(Root->Left)  f(Root->Right)  Show Root->Element |
| DELETE | (On the basis of given value)  **Twist 1 : If both Root->Left and Root->Right is NULL**  Find correct (POSITION, PARENT)  If POSITION is | | | | |
| A Leaf | Having 1 Child | | Having 2 Childs | |
| Delete it straight  away | Bypass POSITION and  set (POSITION's Child)  as (PARENT's Child) | | Find and use POSITIONS's In-Order  (SUCCESSOR, SUCCESSOR\_PARENT)  **Twist 2 : What if POSITION->Right**  **is the In-Order successor** | |
| ***OR*** *Use the Typical Recursive Delete logic, Viz.* | | | | |

**2. Program Analysis: Right Threaded Binary Search Tree**

INSERT

See the Insert function as function of (JIS-ME, JIS-KO, JIS-SE)

(Interpret in Hindi)

**Function of (JISME, JISKO, JISSE)**

Agar JISME khali hoo

JISME = JISKO

Agar JISSE khali naa ho toh

JISME ka Right = JISSE

JISME ka thread = true

Nahito Agar JISKO ka value JISME se kam ho

JISME k left ok JISME k Left me JISKO ko daal k wirasat me JISME ko dekar de do

*JISME ka Left = (JISME ka Left, JISKO, JISME)*

Nahito Agar JISKO ka value JISME se kam ho

Agar JISME ka thread = false ho

JISME ka Right = (JISME ka Right, JISKO, JISSE)

Nahi toh

*NULL, nahi toh upar chata jaiyega*

JISME ka Right = (NULL, JISKO, JISSE)

DISPLAY

Visualize the following in order to write display function and use Recursion.

Root For 1st Cycle

A

Root For 2nd Cycle

G

D

H

E

F

C

B

**3. Program Analysis: AVL Tree**

RIGHT ROTATION = Simple LL Rotation

R

A

R

B

C

A

B

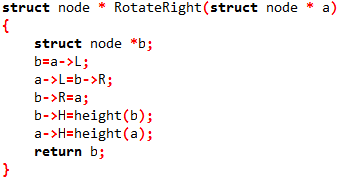
C

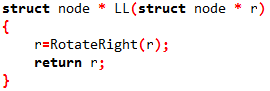
D

D

Balance = 2

Just inserted Node





LR ROTATION = Small Left Rotation + Big Right Rotation

Small Left Rotation

R

B

C

A

B

C

A

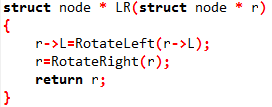
R

R

A

B

C



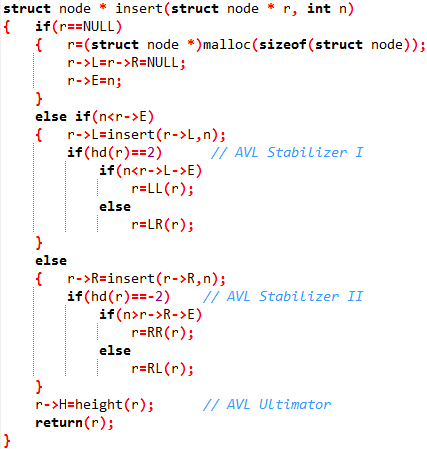
Big Right Rotation

***RR and RL are nothing but mirror images of LL and LR Respectively.***

INSERT

Use Typical Recursive Insert Logic with AVL Stabilizer and AVL Ultimate.

*For Visualization, use*



Balance = 2

R

D

C

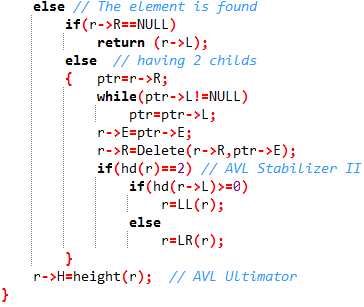
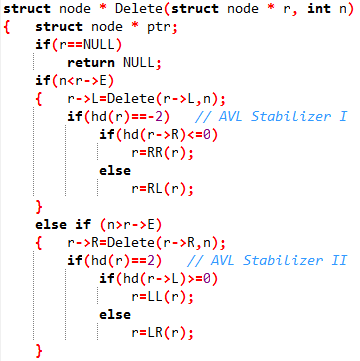
B

A

DELETE

Use Typical Recursive Insert Logic with AVL Stabilizer and AVL Ultimate.

*For Visualization, use*



Balance = 0

Balance = 0

R

A

*And*

L

A

R