# Red Black Tree

M.B.Chandak

www.mbchandak.com

# Basic Characteristics

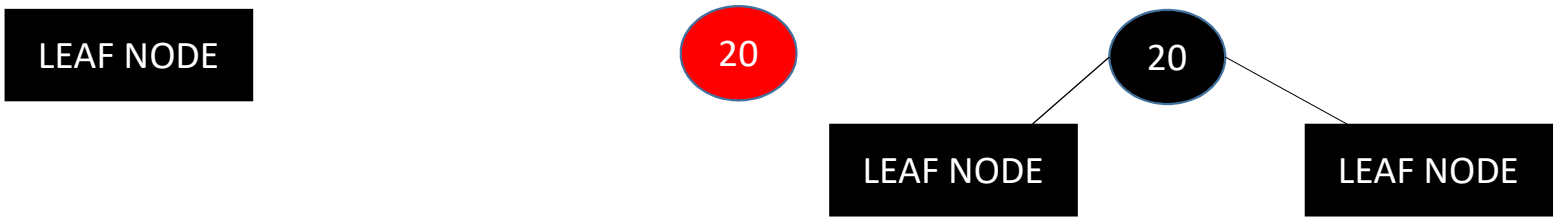
- It a Binary Search Tree [BST]
- The Root node of the tree is black.
- If leaf node is Null then it is considered as Black. [In some operations Null leaf node is also used, hence colouring is important]
- If node is “Red” then its children will be “Black”
- Every path  from a node to a descendant leaf, will contain same number of black nodes.
- Black Height: The number of black nodes in the path from a node, but not including node is called as “Black-height” bh(x)

# Basic Characteristics

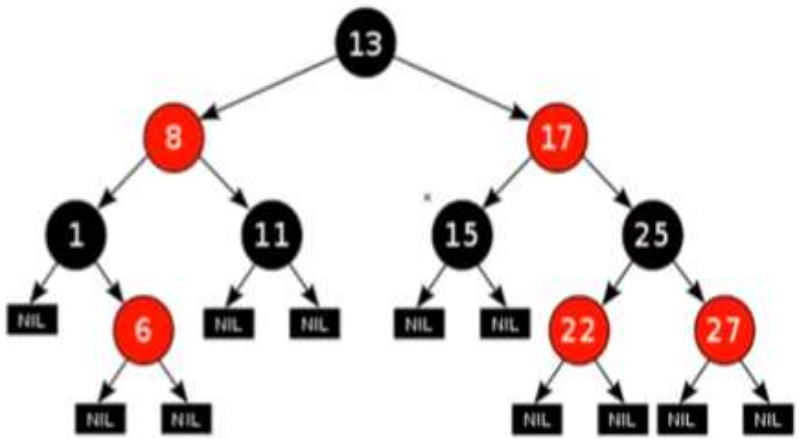
- Rotation: It is an operation required to balance the tree height, and it will be always in opposite direction of insertion.
- Initial colour of any new node is “RED”
- Recoloring operation allows change of colour [RED-BLACK-RED]
- Generally Red-Black recoloring is more required in Tree balancing operation.
- Red nodes can only have “Black” children.

# Terminology


| NODE STRUCTURE | | | |
|---|---|---|---|
| LEFT POINTER | DATA | COLOR | RIGHT POINTER |

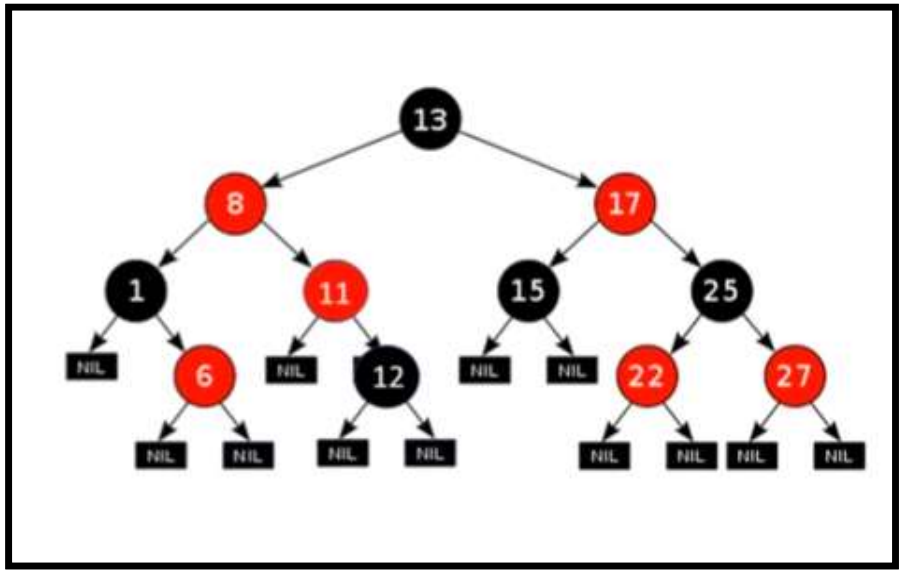


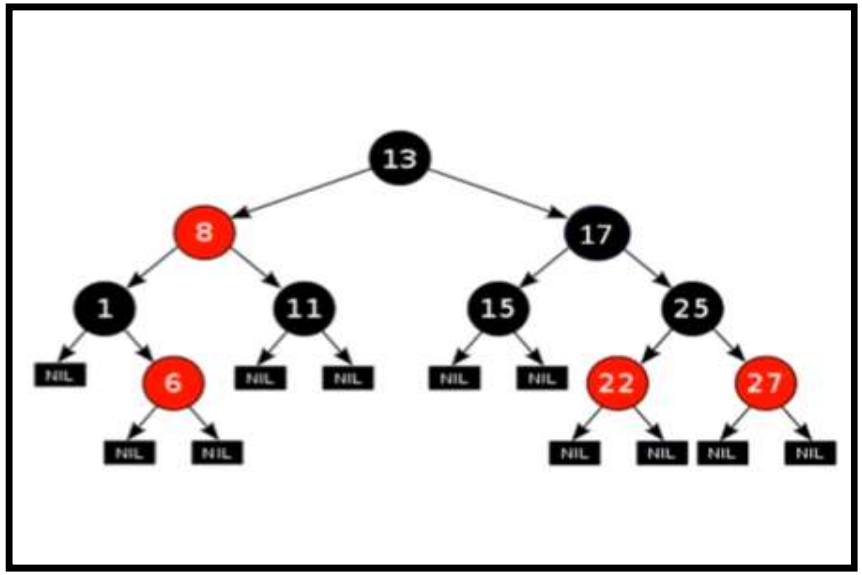
# Sample Examples




- It is a BST
- Each node -> Red/Black
- Head node -> Black
- Nil nodes -> Black
- Red Nodes -> only black children
- # black nodes to leaves-> same

# Sample Examples



# Insertion-Rules and implementation

- Let new node to be inserted in RB Tree denoted as “z”
- After performing “rotation or recoloring operation” new node (z) will be lowest node in violation path.
- Priority is given to parent node, during rotation. [If the left sub-tree needs to be shifted to right part or right-sub-tree is to be shifted to left part]
- **Grand Parent-Parent-Child-Sibling [Uncle relationship]**

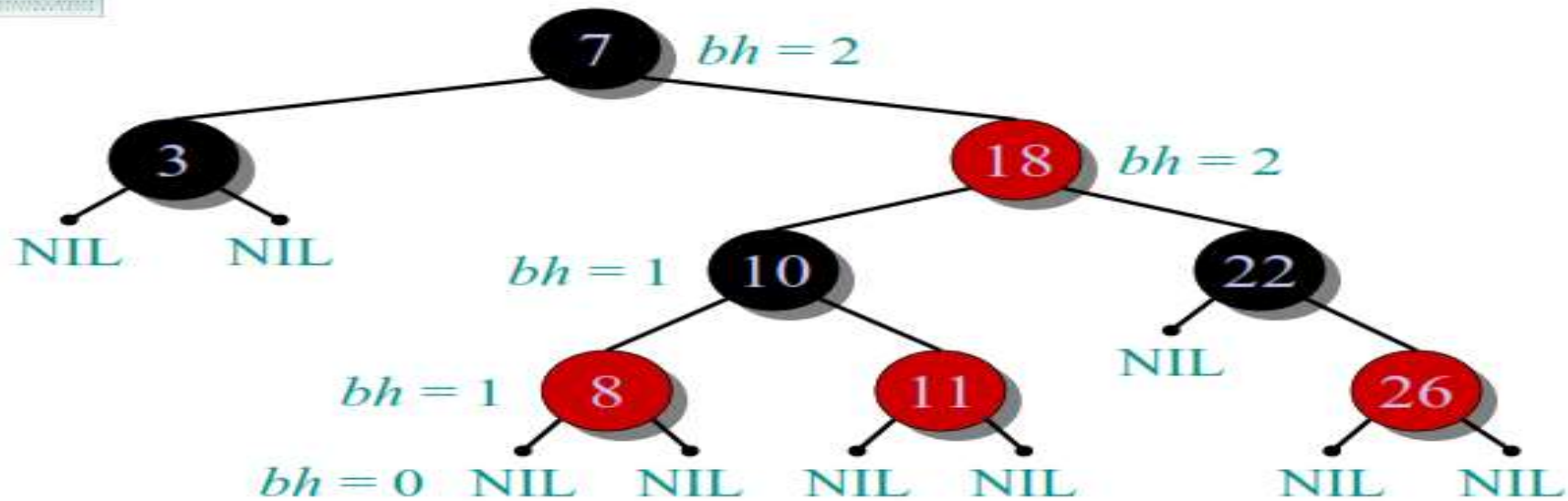
# Creation and Insertion Algorithm

- **Case -1:**
- **"Z" is inserted as first node, then it is first considered as "red" and then coloured as "black"**
- **Case – 2: Dependent on Colour of Sibling of parent [UNCLE]**
- **If (Z.U = Red)**
  - **Change the colour of Z.P, Z.U and Z.GP**
- **Case – 3:**
  - **Z.U = Black and new node "Z" makes a triangle with Z.P, Z.U and Z**
  - **Rotate Z.P in opposite direction**

# Insertion

- **Case – 4:**
  - **Z.U = Black and new node "Z" makes a line between Z.P, Z**
  - **Rotate Z.GP in opposite direction**
  - **Exchange the colours of Grand Parent and Parent.**

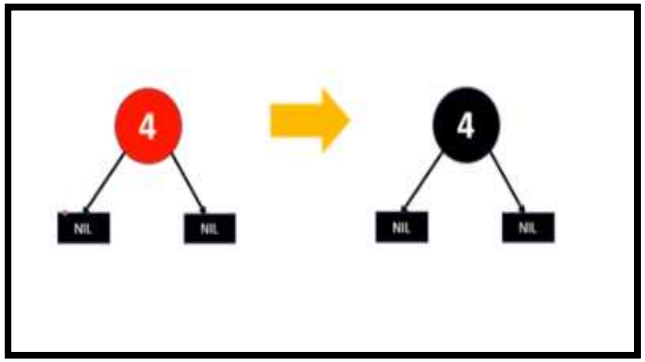
# Bh(x)=Black height of node “x”



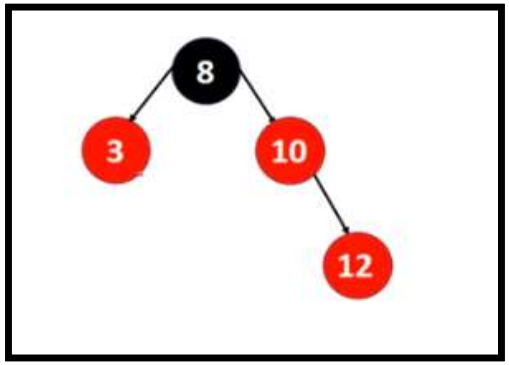

4. All simple paths from any node $x$ to a descendant leaf have the same number of black nodes $= black\text{-}height(x)$.

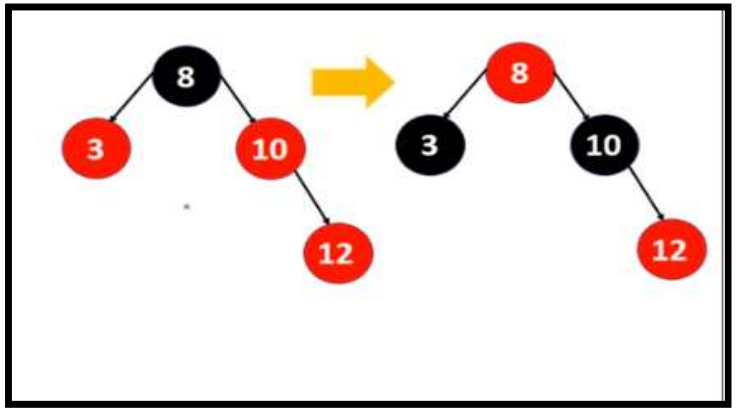
# Sample Example: CASE-1

## CASE-1

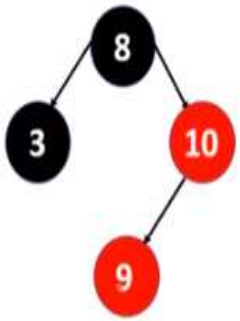


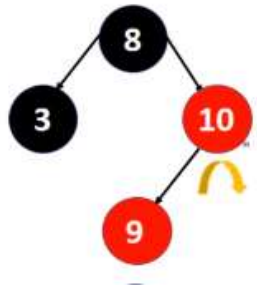

## CASE-2



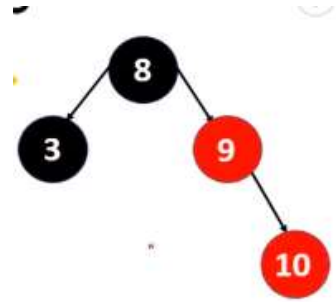



# Sample Cases: Z.U=Black and Triangle

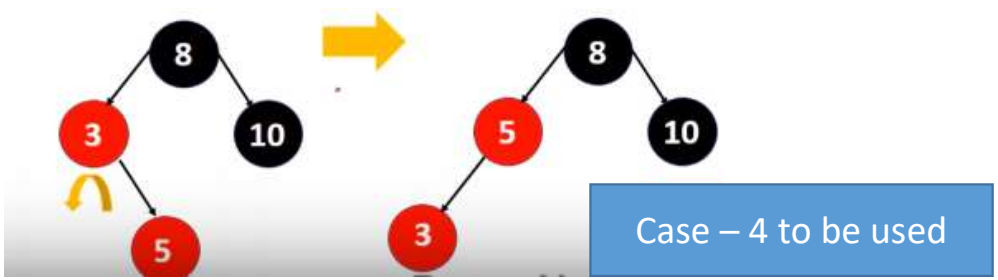



Z.U=Black Triangle is formed. Rotate in opposite direction

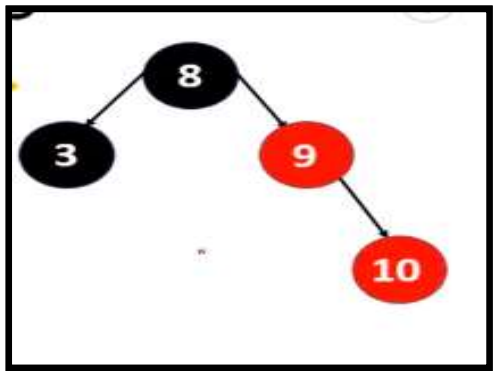


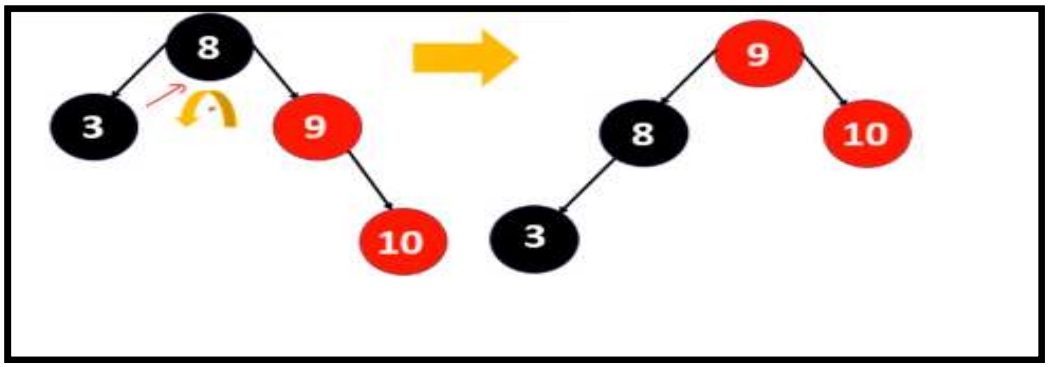

If violation exists: Check CASE-4

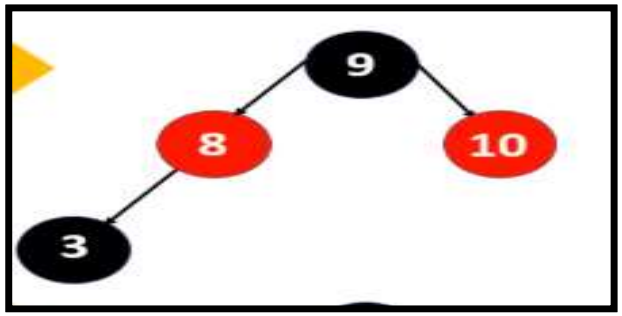



Case – 4 to be used

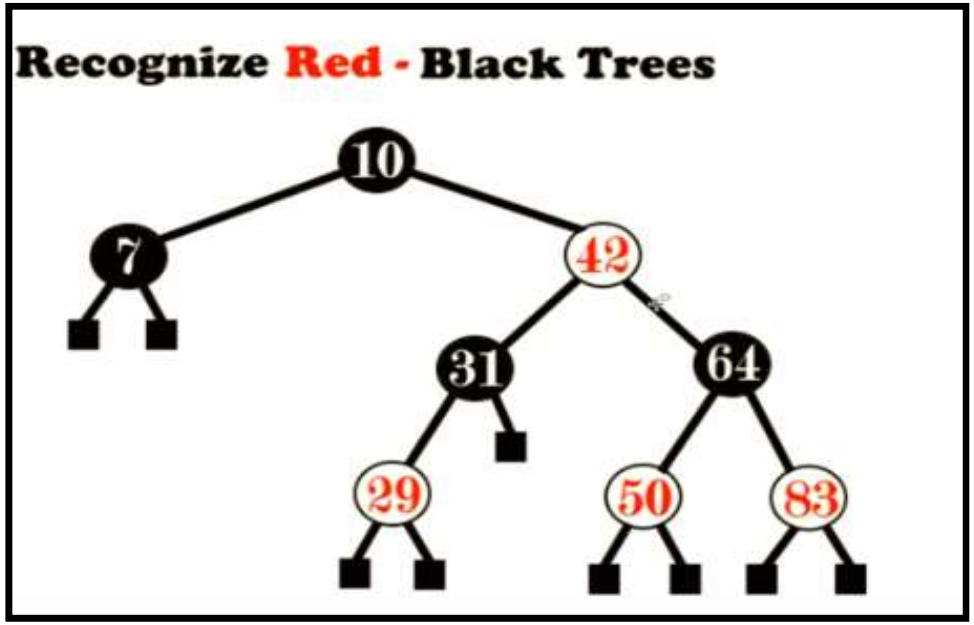
# Sample cases: Case-4

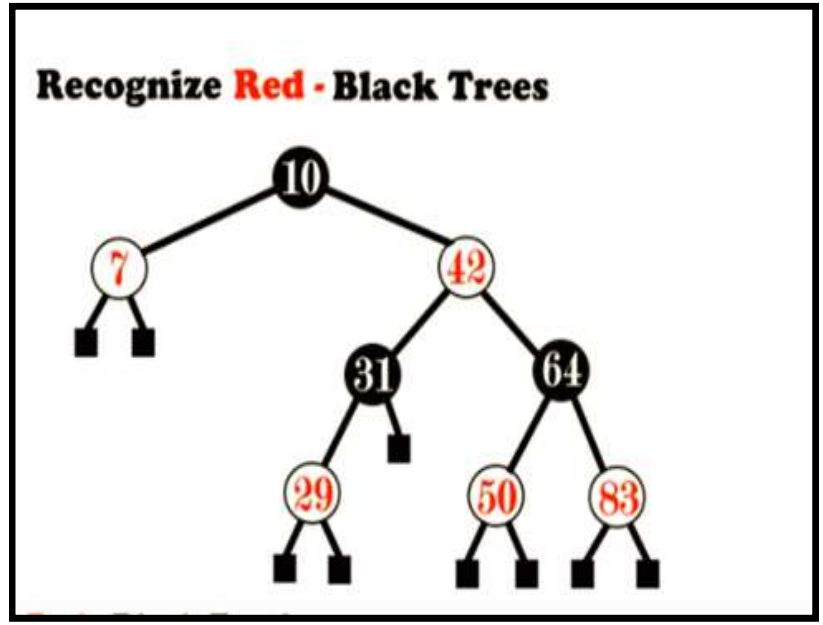


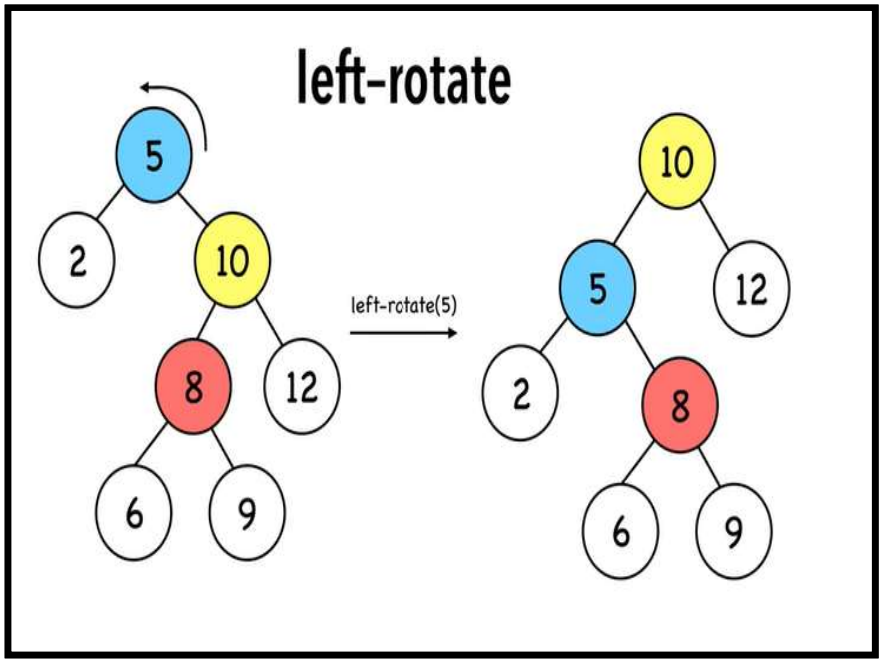



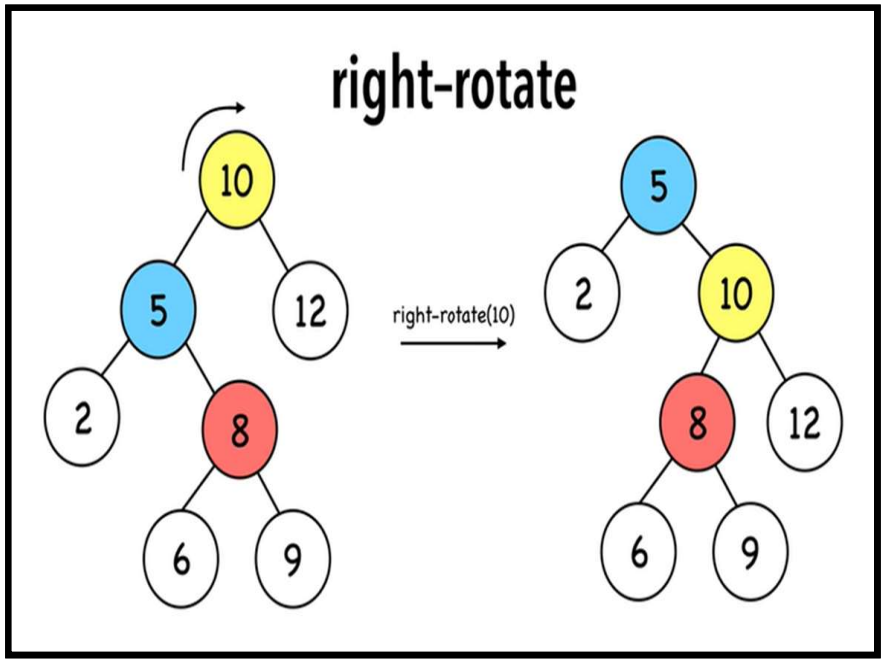
# Justify: Red/Black Tree [T/F]



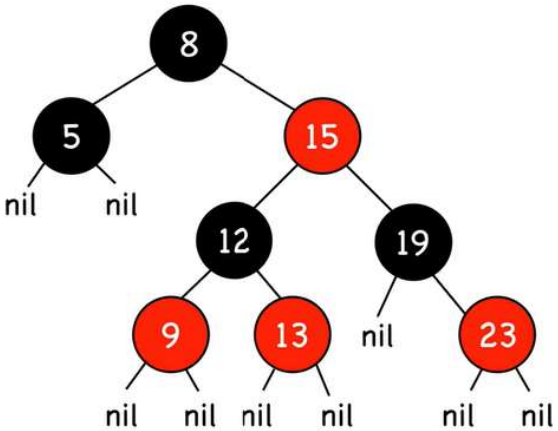



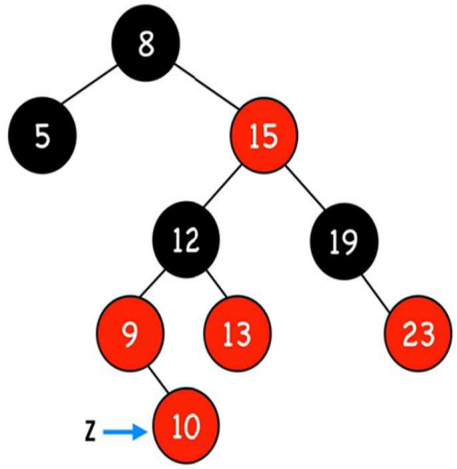
# Left & Right Rotate [O(1)]

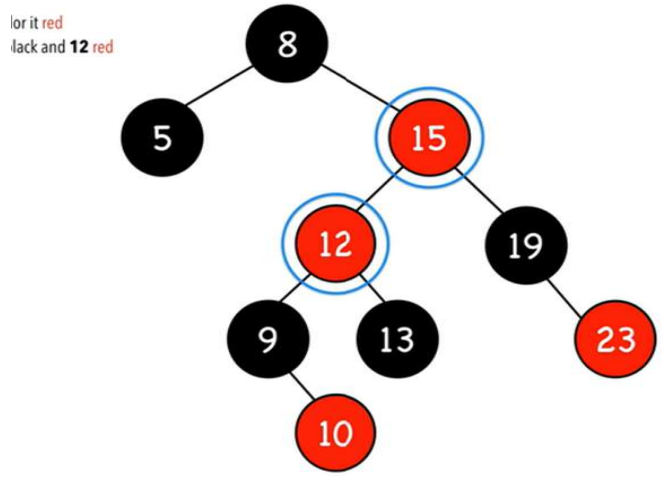


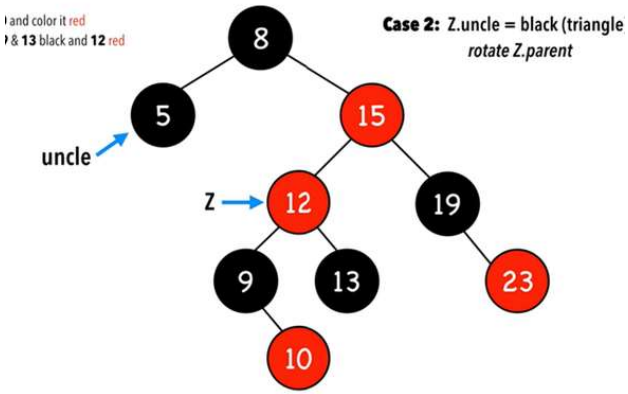



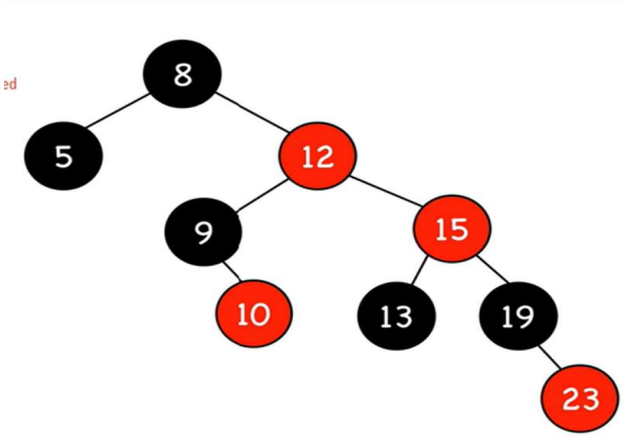
# Example-1: Create and Insert

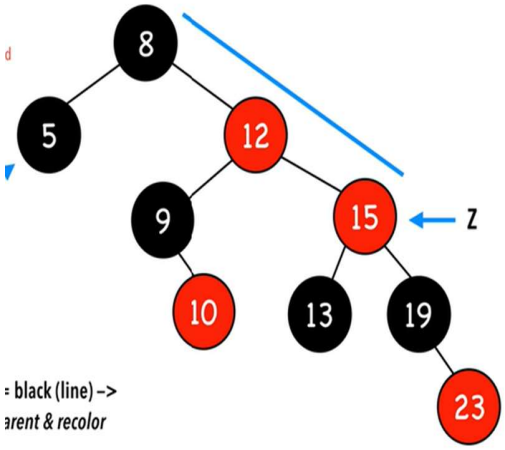


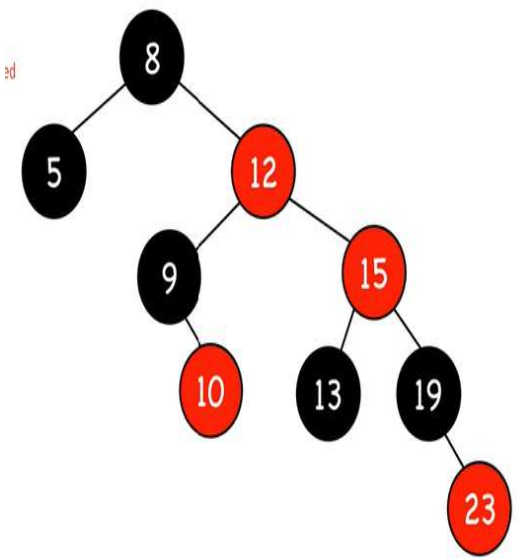



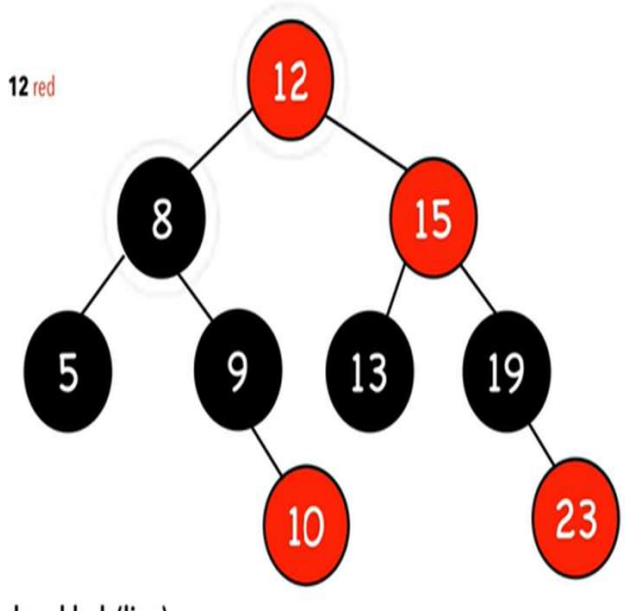



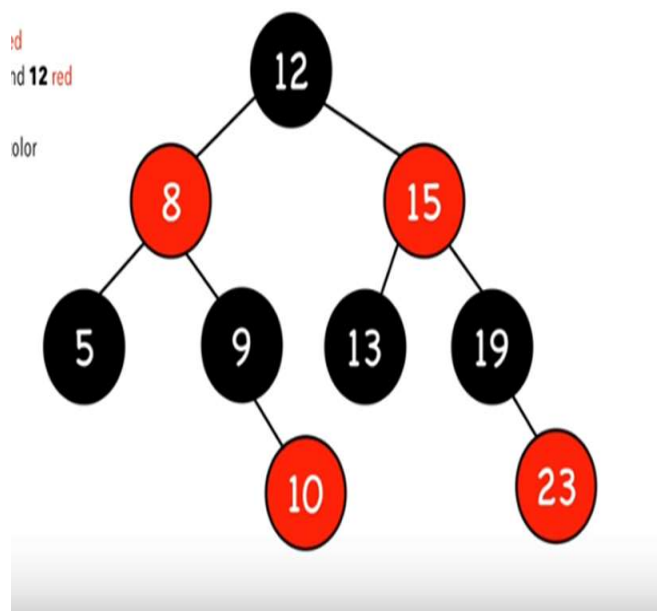



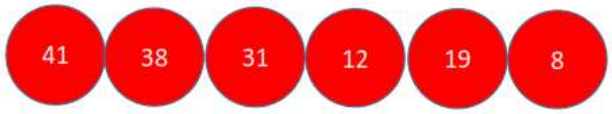




8
5
12
9
15
10
13
19
23

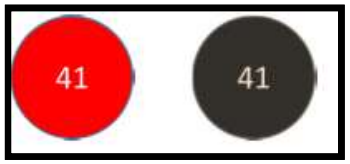

12 red
12
8
15
5
9
13
19
10
23


12
8
15
5
9
13
19
10
23

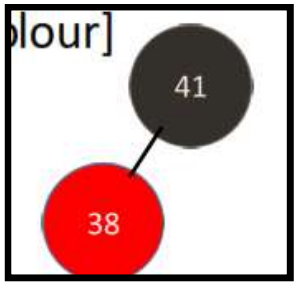
# Example-Insertion and Creation of RB-Tree



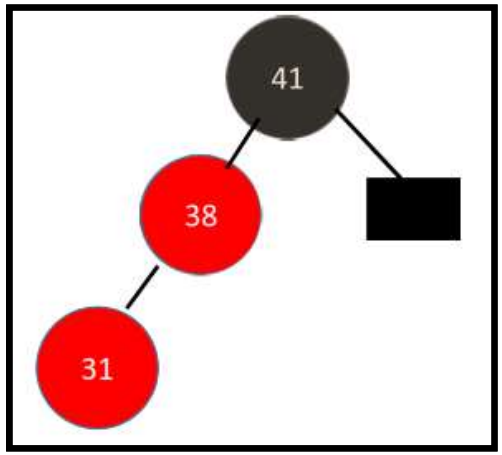

Step-1: Select root node, insert and change colour



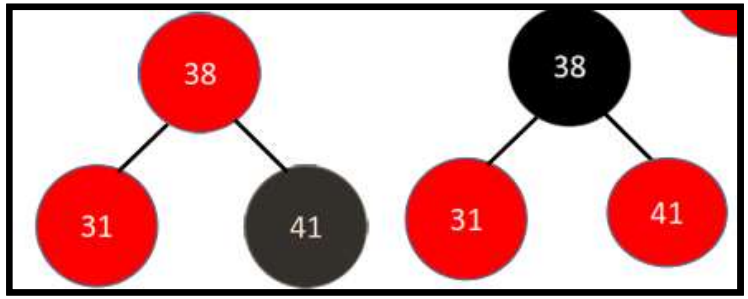

Step-2: Insert 38 on Left of 41, no change in colour




Step-3: Insert 31 on Left of 38 with Red colour

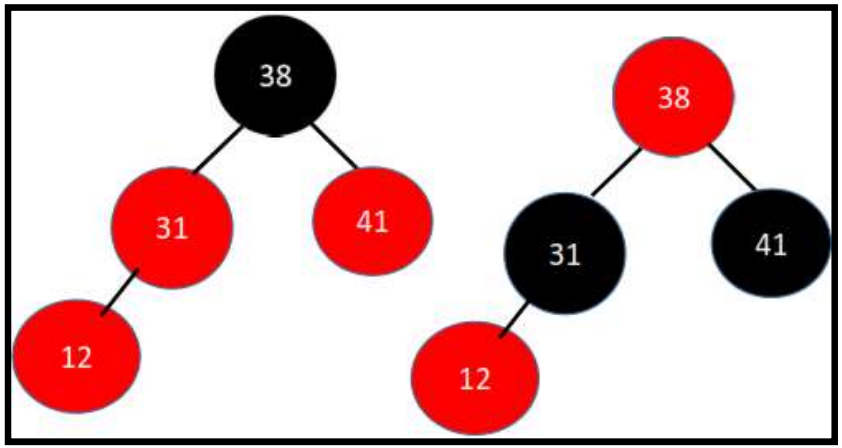


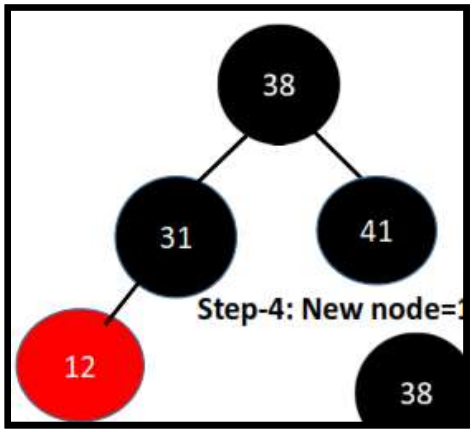

Rotate Z.P in opposite direction of Z
Exchange colour of GP and P



# Example-Insertion and Creation of RB-Tree

Insert 12: Left of 31, in red colour

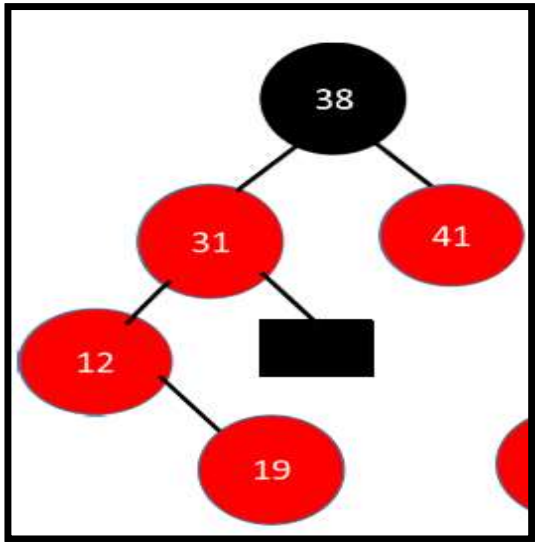


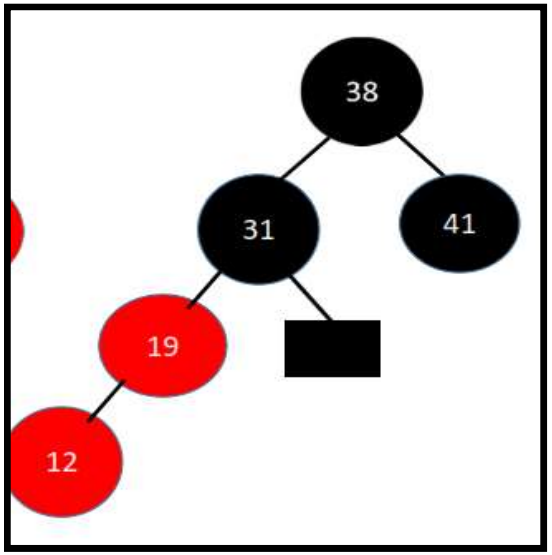



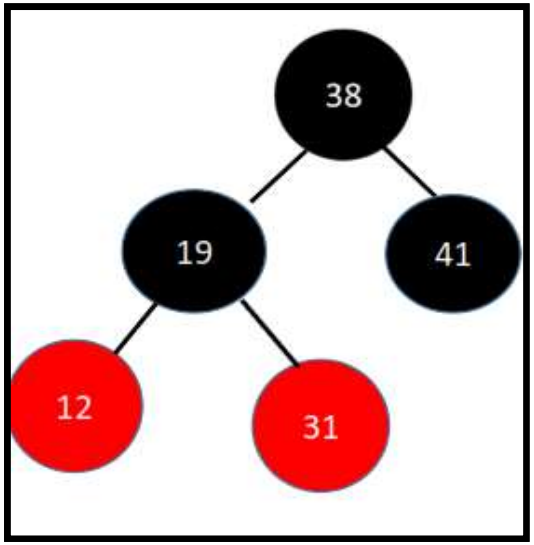

**Step-4: Z.U=RED, change colour of, PARENT, UNCLE, GP**
**At the end of Root is RED-Directly change it to black**

# Example-Insertion and Creation of RB-Tree

Insert 19: Right of 12, in red colour



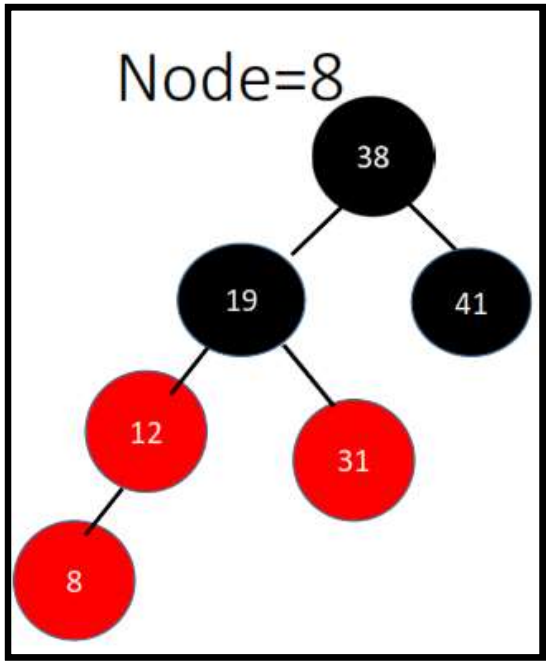



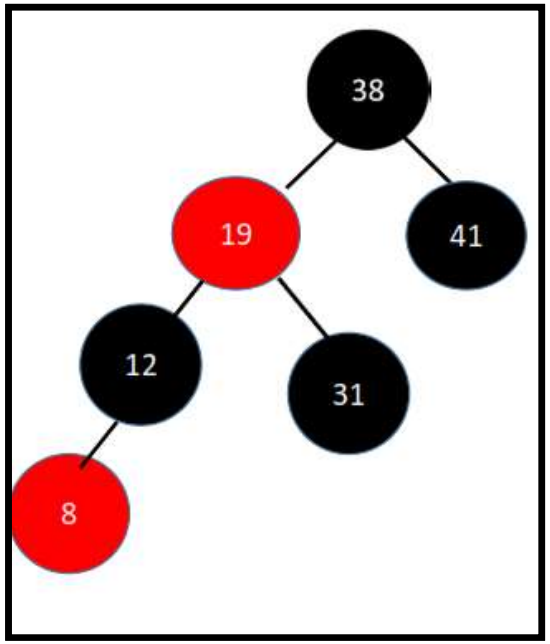

Z.Uncle is black, Triangle is formed, Rotate Z.Parent in opposite direction of Z

Case 4 – Rotate Z.GP in opposite direction and Exchange the colour of GP and P

# Example-Insertion and Creation of RB-Tree

Insert 8: Left of 12

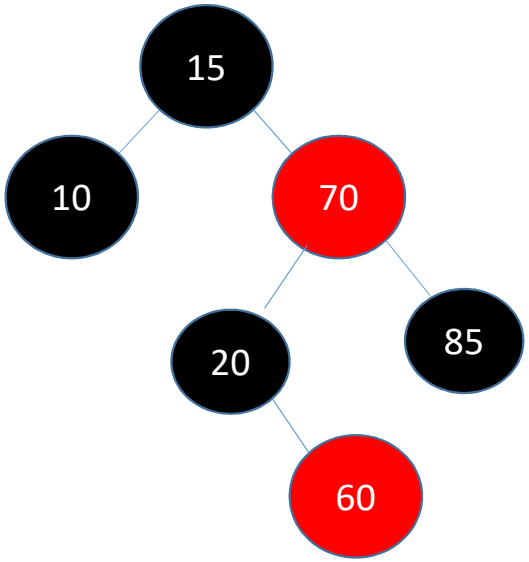



Z.Uncle=Red, Exchange colour of Z.P, Z.U and Z.GP

# Home Assignment

- Example:1
- 8,18,5,15,17,25,40
- Example-2
- 10,85,15,70,20,60,30,50,65,80,90,40,5,55

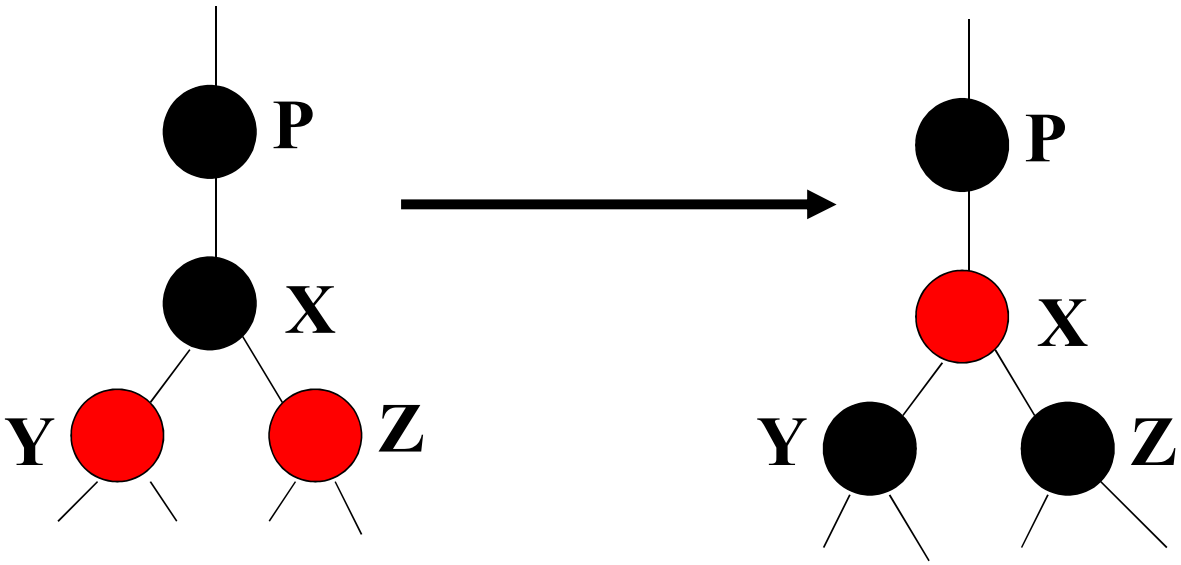


# Insertion-Algorithm

# Top-Down Insertion: RB Trees

- In **B-Up insertion**, "ordinary" BST insertion was used, followed by correction of the tree on the way back up to the root. [LEAF – ROOT]
- In **T-Down insertion**, the corrections are done while traversing down the tree to the insertion point.
- When the **actual insertion is done, no further corrections** are needed, so no need to traverse back up the tree.
- Therefore, the goal of T-D insertion is to traverse from the root to the insertion point in such a way that RB properties are maintained, and at the insertion point, the uncle is Black

# Case 1 – X's Parent is Black

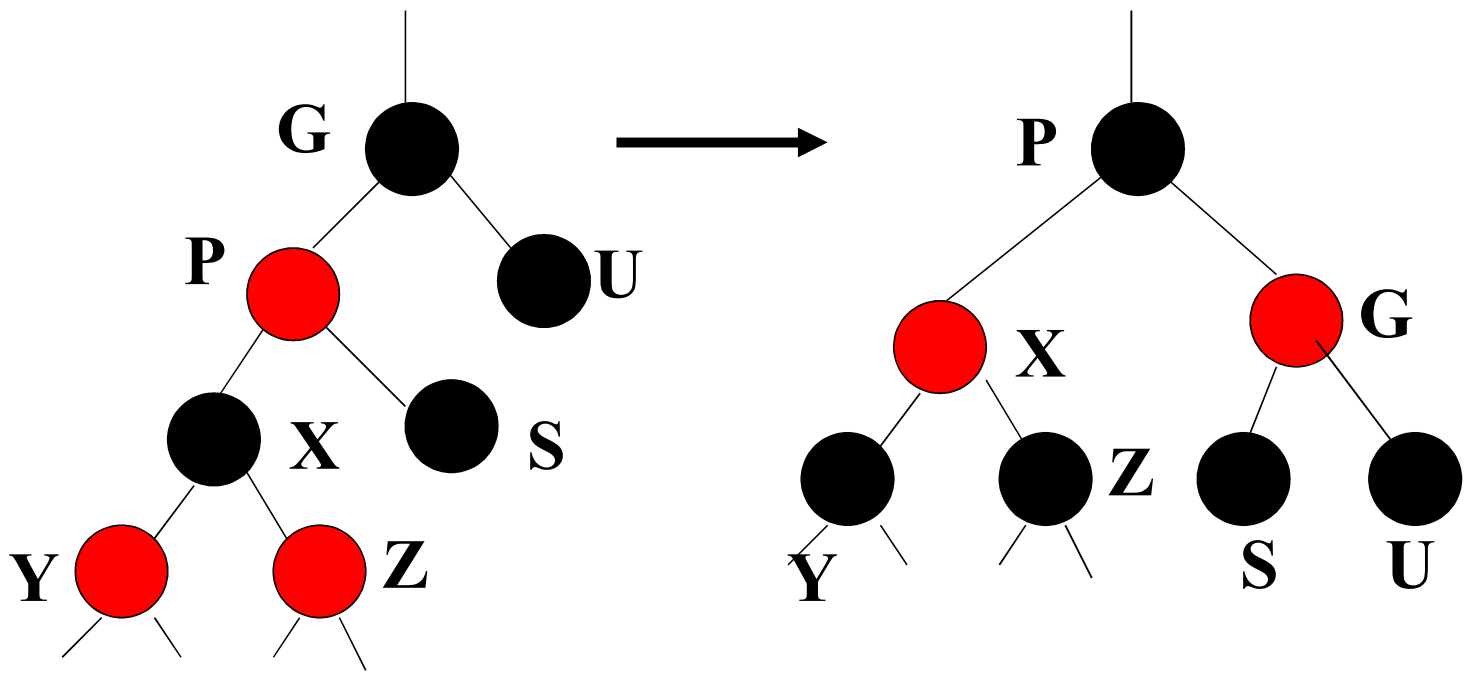



Just recolor and continue down the tree

# Case 2

- **X's Parent "P" is Red** (so Grandparent is Black) and X and P are both left OR right children of Grandparent.
  - Rotate P around G
  - Color P black
  - Color G red
- Note that X's uncle, U, must be black because it (a) was initially black, or (b) would have been made black when we encountered G (which would have had two red children -- X's Parent and X's uncle)

# Case 2 diagrams

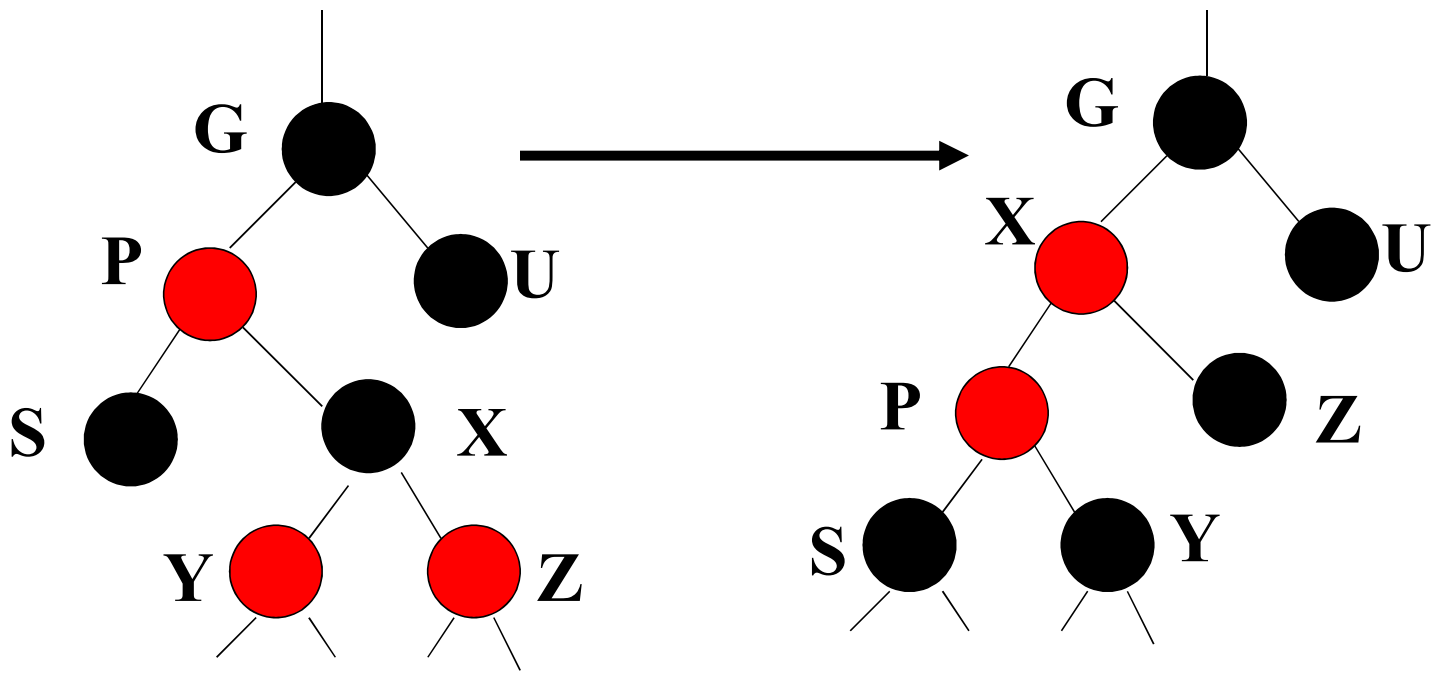



Rotate P around G. Recolor X, Y, Z, P and G

# Case 3

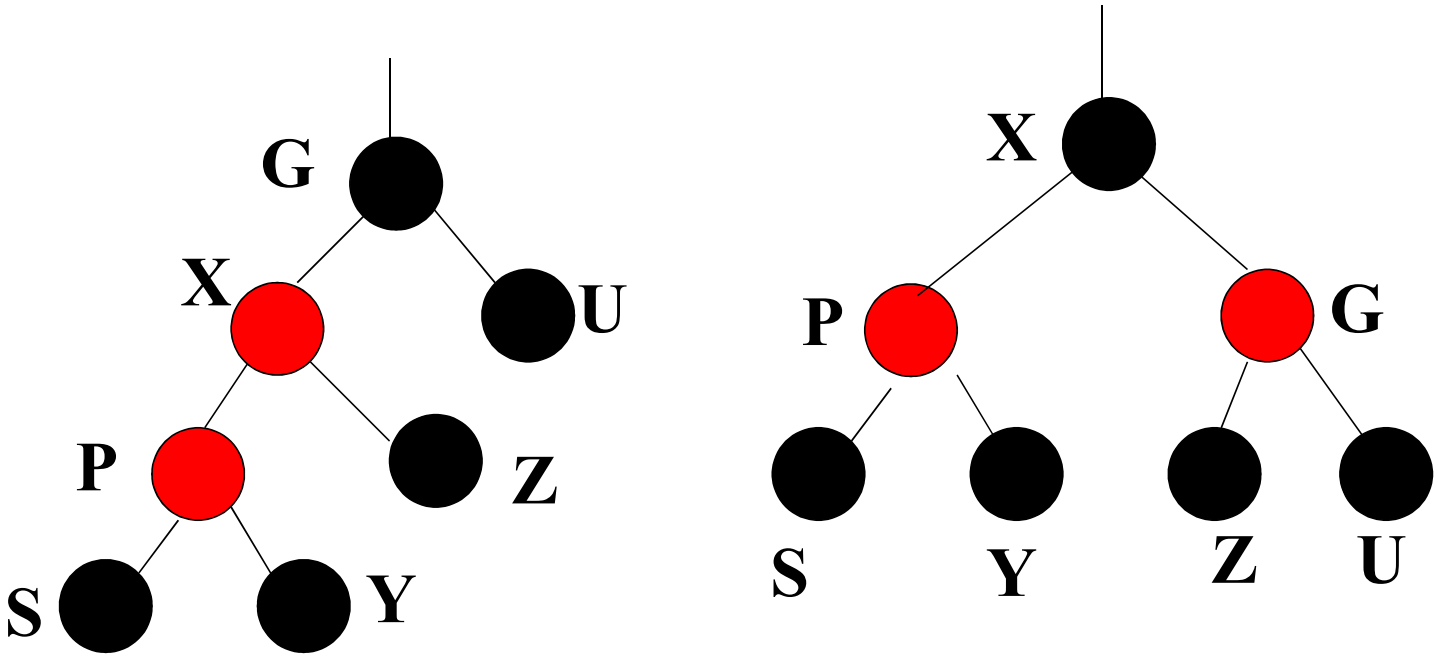
- **X's Parent is Red** (so Grandparent is Black) and X and P are opposite children
  - Rotate P around G
  - Color P black
  - Color G red
- Again note that X's uncle, U, must be black because it (a) was initially black, or (b) would have been made black when we encountered G (which would have had two red children -- X's Parent and X's uncle)

# Case 3 Diagrams (1 of 2)




Step 1 – recolor X, Y and Z. Rotate X around P.

# Case 3 Diagrams (2 of 2)



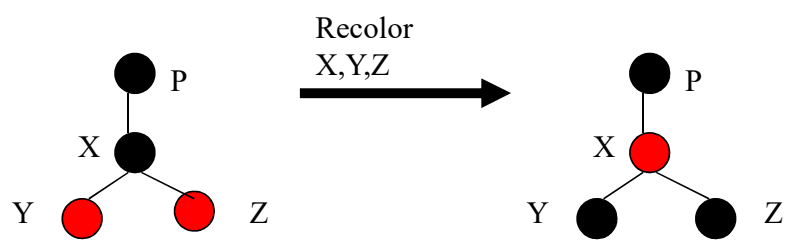

Step 2 – Rotate X around G.  Recolor X and G

# Top-Down Insert Summary

Case 1

P is Black

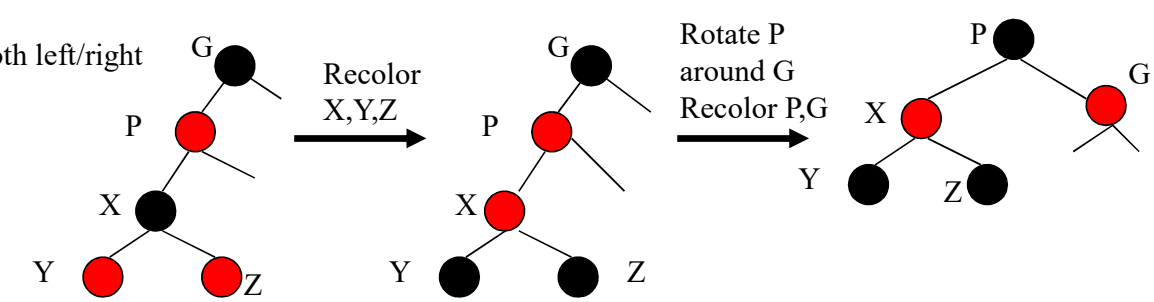
Just Recolor




Case 2

P is Red

X & P both left/right

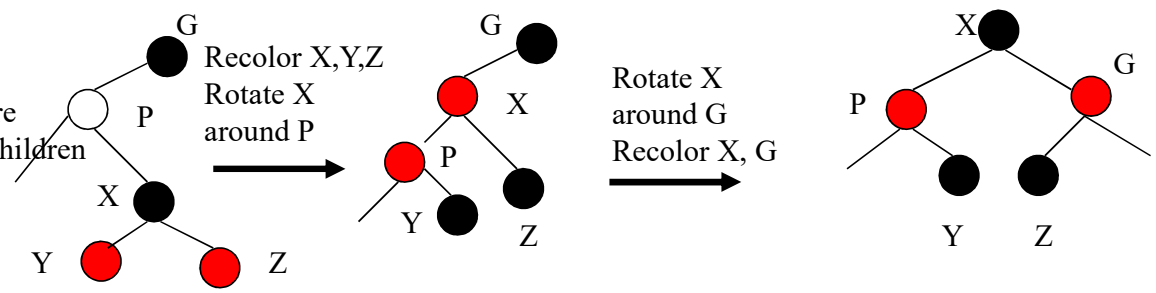



Case 3

P is Red

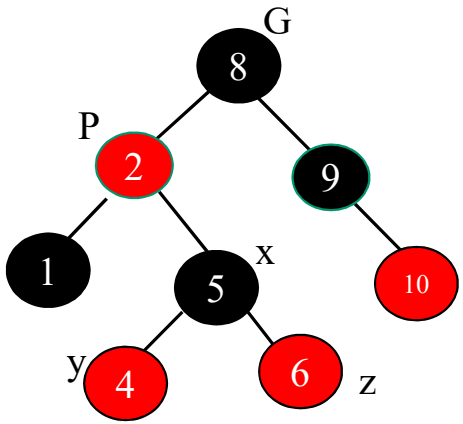
X and P are

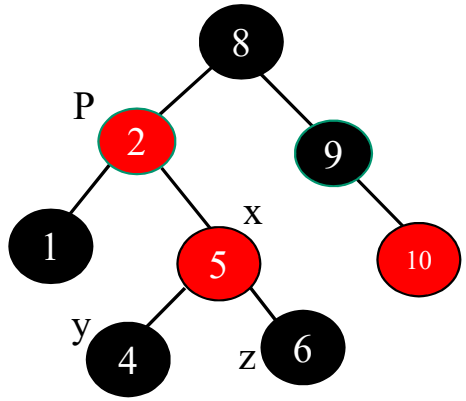
opposite children

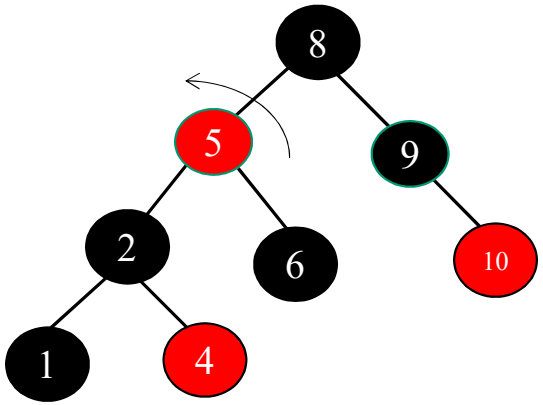


# Top down Insertion: RB Tree [case 2.2]

**Inset Node 3**

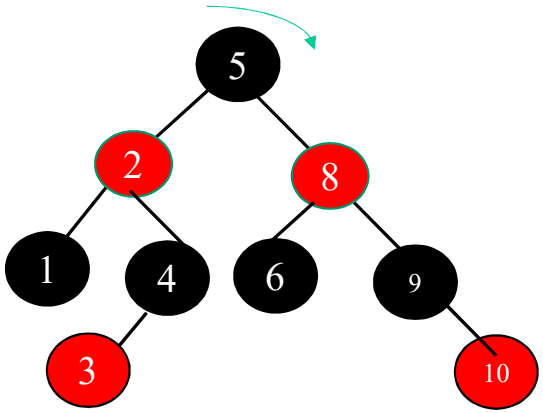


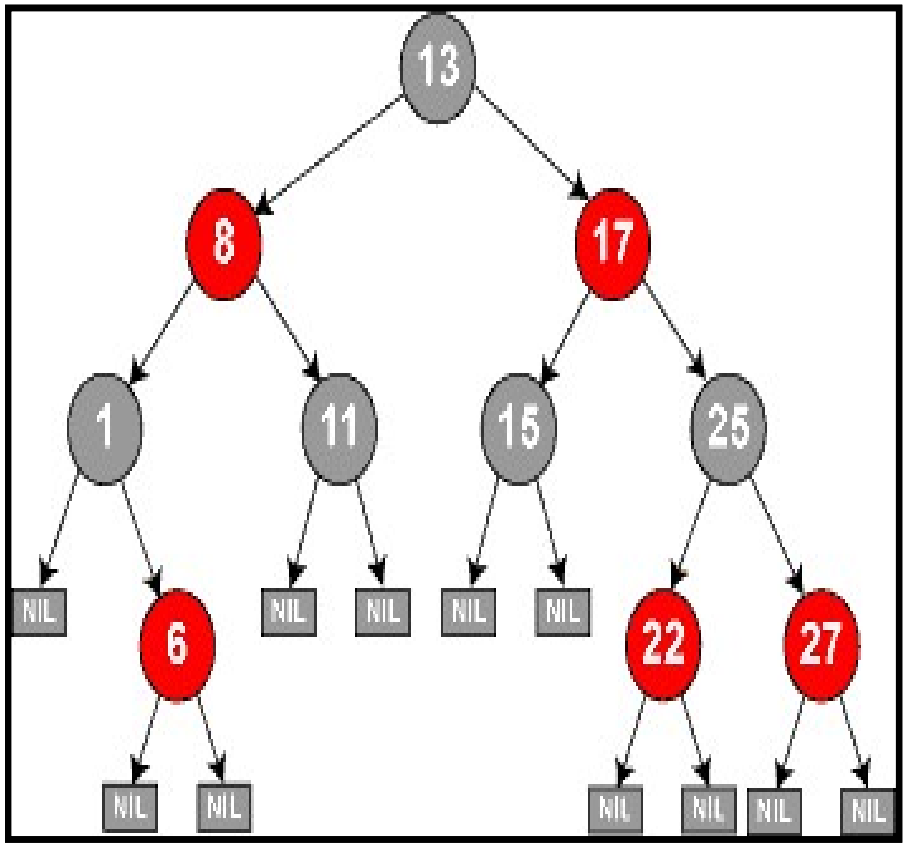



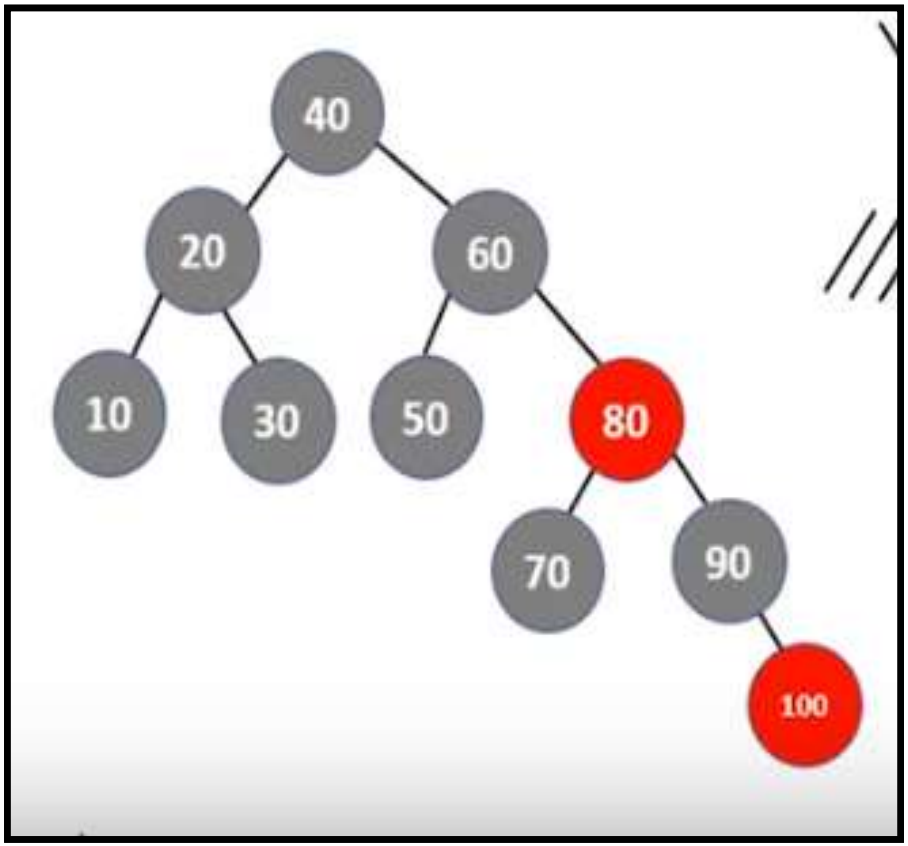

**P is RED and X and P are opposite**

- **Recolour X, Y, Z**
- **Rotate X around P**
- **Rotate X around G**
- **Recolour X and G**

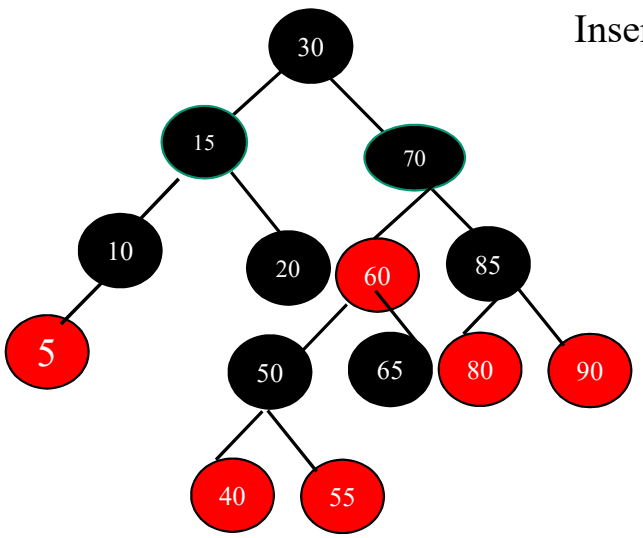


# Delete Node: RB Tree Example



# RB Tree insertion: Case 2.1




Insert 45

P=Red
X and P are right or left branch
Recolour X,Y,Z
Rotate P around Node G
Recolour P and G