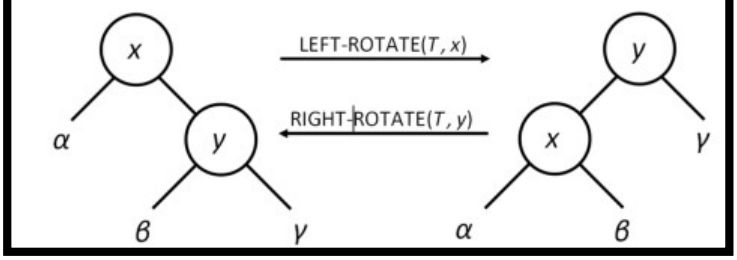
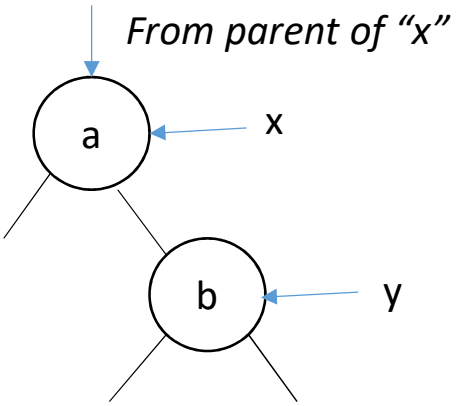


Node Structure

Left	Parent	Data	Right

Example:
x.Left = y
y.Parent = x



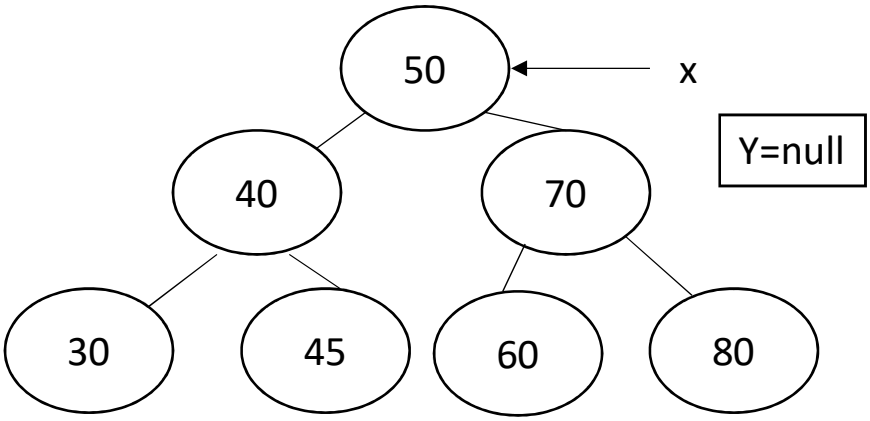

```
y=x.right
x.right= y.left          [check β]
y.left.parent=x          [node structure]
y.parent=x.parent
If(x.parent=NULL) then
   T.root=y
    else
       if(x=x.parent.left)
            x.parent.left=y
        else
             x.parent.right=y
y.left=x
x.parent=y
```

Change left=right and right=left to perform

Part-1: To find the location of insertion.
Part-2: To change the colour of nodes [if required]

Let “Z” be the node to be inserted.
Z.Left=Null
Z.Right=Null
Z.Colour=Red

Now it is required to traverse the tree from “root” and find the suitable position for
Inserting node “Z” in the tree.




```
While (x != null) do
{
  y=x  //change y before changing x
  if(z.data < x.data)
  {
     x = x.left
   else
     x =x.right
   }
   z.parent=y
   if(y=null) // no tree exist
     t.root=z
   if(z.data < y.data)
     y.left=z
   else
     y.right=z
--- Node “Z” pointers are initialized, as above
```

Case – 1: Dependent on Colour of Sibling of parent [UNCLE]
If (Z.U = Red)
Change the Colour of Z.P, Z.U and Z.GP
Case – 2:
Z.U = Black and new node “Z” makes a triangle with Z.P, Z.U a
Z
Rotate Z.P in opposite direction
Case – 3:
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.

While (Colour[p[z]]) = Red) do
{
  if (p[z] = left(p[p[z]]) then
  { ***Case 1***
    y = right[p[p[z]]] //Mark y as uncle
    if(colour[y]=red) then //recolour
    {
      colour[p[z]] = Black
      colour[y] = Black
      colour[p[p[z]]] = Red
      z=p[p[z]]
    } ***//Case 2***
    else
    {
      if (z=right[p[z]]) then
      {
        z = p[z]
        Left-rotate(T,z)
      } ***Case 3***
      colour[p[z]] = Black
      colour[p[p[z]] = Red
      Right-rotate(T, p[p[z]])
    }
  else [Same only interchange right-left]
Again while loop is checked, if false
Colour[root[T]] = Black
//End of algorithm