Rotation is a six-pointer maneuver. There are four nodes to concern ourselves with: the node to be rotated (x) , the parent node (p) , the grandparent node (gp) (if there is one) and the zigzag child (z) (if there is one).

1. x's parent needs to point to gp.

2. If p is a right child of gp, gp's right needs to point to x.
   If p is a left child of gp, gp's left needs to point to x.
   If gp does not exist, then the root needs to point to x.

3. p's parent needs to point to x.

4. If x is a left child of p, x's right needs to point to p.
   If x is a right child of p, x's left needs to point to p.

5. If x is a left child of p, p's left needs to point to z.
   If x is a right child of p, p's right needs to point to z.

6. If z exists, z's parent needs to point to p.

Some methods you will need to have:

bool getColor (Node *n); //returns black if n is NULL, otherwise returns the color of n.

Node * Node::getSibling (Node *n, Node *p); //returns the child of p that isn't n.

Node *Node::getDirect (); //if I am the left child of my parent, returns my left child. If I am the right child of my parent, returns my right child.

bool Node::isDirect(); //returns true if I am the direct child of my parent, false if I am a zigzag child.

void AddProcess (Node *x) processes rules 3 to 7 on x. If the rules say to RESTART, you can recursively call this method on the appropriate value of x.

void DelProcess (Node *x, Node *p) processes rules 4 to 11 on x and p. If the rules say to RESTART, you can recursively call this method on the appropriate values of x and p.