# COP 5536 Programming Project

*Name: Mayank Wadhawan*
*UFID - 59148122*

I created this project in Java and I used java compiler. There is one java file titled bbst.java. Each node in the Red-Black tree is of type RedBlackTreeNode. This is defined as a class in bbst.java.

**RedBlackTreeNode contains fields :-**

1. **parent** - Pointer to parent node
2. **color** - Color of node
3. **left** - Left node pointer
4. **right** - Right node pointer
5. **totalLeft** - Total elements on left side of node
6. **totalRight** - Total elements on right side of node
7. **key** - Stores the key
8. **count** - Value of count

In this project, initially I have set parent, left and right pointers of new node to sentinel(nil). After insertion, fixup is required to maintain Red-Black tree property. However, after deletion, fixup may be required.

**Below are the list of functions along with their brief description:-**

- **public static void main(String[] args)**
  This is the main method and it drives the project.
- **public void insert(int id, int count)**
  Used to insert a new node in Red-Black tree
- **private void fixInsert(RedBlackTreeNode nodeToFix)**
  Performing fixup after insertion
- **public RedBlackTreeNode findSuccessor(RedBlackTreeNode x)**
  Finding next node greater than current node
- **public void lowestNext(int theID)**
  Finding node with smallest id
- **public void greatestPrev(int theID)**
  Finding node with greatest id
- **public void deleteNode(int key)**
  Deleting a node from Red-Black tree
- **public RedBlackTreeNode findNodeInTree(int key)**
  Finding node with a key in Red-Black tree

- **public RedBlackTreeNode minimumInTree(RedBlackTreeNode node)**
  Finding node with minimum key in Red-Black tree
- **public List<Integer> findBiggerElement(int key, Integer maxReturned)**
  List of elements greater than key
- **private boolean checkIfNodeIsNull(RedBlackTreeNode node)**
  Checking if node is sentinel
- **public RedBlackTreeNode maximumInTree(RedBlackTreeNode node)**
  Finding node with maximum key in  Red-Black tree
- **private void fixAfterDeletion(RedBlackTreeNode nodeToDelete)**
  Performing fixup after Deletion
- **public void increaseCount(int theID, int count)**
  Increase count of id by count. If the id is not in the tree, then we insert it. I will also display the count after increasing it.
- **public void decreaseCount(int theID, int count)**
  Decrease count of id by count. If count is less than or equal to zero, then I will remove the id from Red-Black tree. I will also print the count after decreasing. If the id is removed or not present, then I will display 0.
- **public void findCount(int theID)**
  Print count of ID. If id is not present, I will display 0.
- **public void checkInRange(int ID1, int ID2)**
  Finding sum of count of id's b/w id1 and id2.
- **private void rotateLeft(RedBlackTreeNode nodeToRotate)**
  Performing left rotation and updating children information afterwards.
- **private void rotateRight(RedBlackTreeNode nodeToRotate)**
  Performing right rotation and updating children information afterwards.