Student Names-IDs:	Elsayed Akram Elsayed-16 Hassan Abdl-Hameed Hassan-25
Lab Title:	Implementing Red Black Tree & Treemap interface

Computer and Systems Engineering Department

1 Time analysis of Functions

n : the number of nodes in the tree

1.1 Red Black Tree

1. getRoot(): $\mathcal{O}(1)$
2. isEmpty(): $\mathcal{O}(1)$
3. clear(): $\mathcal{O}(1)$
4. search(T): $\mathcal{O}(\log n)$ as it uses method findNode(T) that has also run time $\mathcal{O}(\log n)$
5. contains(T): $\mathcal{O}(\log n)$ as it uses method findNode(T)
6. insert(T, V): consists of 3 steps
   (a) ordinary BST insert for balanced tree which its worst run time is $\mathcal{O}(\log n)$
   (b) the new node is already colored Red so the coloring run time is $\mathcal{O}(1)$
   (c) fixing the properties of red black tree using method redRed(node) which its worst run time is $\mathcal{O}(\log n)$

   so the total worst run time $\mathcal{O}(\log n)$
7. delete(T): consists of 4 steps
   (a) using findNode(T) to get the node to be deleted in $\mathcal{O}(\log n)$
   (b) in case it's an internal node find its replacement using findMin(node) $\mathcal{O}(\log n)$
   (c) remove the node and replace it with a new node by method removeAndReplace(node, node) in $\mathcal{O}(1)$
   (d) fixing the properties of red black tree using method doubleBlack(node) which its worst run time is $\mathcal{O}(\log n)$

   so the total worst run time $\mathcal{O}(\log n)$
8. leftRotate(node) & rightRotate(node): $\mathcal{O}(1)$

2 Tree Map

1. ceilingEntry(T): $\mathcal{O}(\log n)$ as it uses method findNode(T)
2. ceilingKey(T): $\mathcal{O}(\log n)$ as it uses method ceilingEntry(T)
3. clear(): $\mathcal{O}(1)$ as it uses the red black tree clear() method
4. containsKey(T): $\mathcal{O}(\log n)$ as it uses the red black tree contains(T) method
5. containsValue(V): $\mathcal{O}(n)$ as it uses values() method

6. entrySet(): $\mathcal{O}(n)$ as it uses method InorderTraversal(node) which its run time is $\mathcal{O}(n)$
7. firstEntry(): $\mathcal{O}(\log n)$
8. firstKey(): $\mathcal{O}(\log n)$ as it uses firstEntry()
9. floorEntry(T): $\mathcal{O}(\log n)$ as it uses findNode(T)
10. floorKey(T): $\mathcal{O}(\log n)$ as it uses floorEntry(T)
11. get(T): $\mathcal{O}(\log n)$ as it uses the red black tree search() method
12. headMap(T) & headMap(T, boolean): $\mathcal{O}(n)$ as it uses method InorderTraversal(node)
13. keySet(): $\mathcal{O}(n)$ as it uses method InorderTraversal(node)
14. lastEntry(): $\mathcal{O}(\log n)$
15. lastKey(): $\mathcal{O}(\log n)$ as it uses lastEntry()
16. pollFirstEntry(): $\mathcal{O}(\log n)$ as it uses firstEntry() and remove(T)
17. pollLastEntry(): $\mathcal{O}(\log n)$ as it uses lastEntry() and remove(T)
18. put(T, V): $\mathcal{O}(\log n)$ as it uses the red black tree insert(T, V) method
19. putAll(map): $\mathcal{O}(n \log n)$ as it uses the red black tree insert(T, V) method n times
20. remove(T): $\mathcal{O}(\log n)$ as it uses the red black tree delete(T) method
21. size(): $\mathcal{O}(n)$ as it uses method InorderTraversal(node)
22. values(): $\mathcal{O}(n)$ as it uses method InorderTraversal(node)