PROJECT 1 REPORT
COP5536 : ADVANCED DATA STRUCTURES

Name: Akshitha Adepu
UFID: 12589466
Email_id: akshitha.adepu@ufl.edu

The code contains five classes:

1. **RideRequest**: A class that represents a ride request with attributes for ride identifier, estimated cost and time taken.

2. **MinHeapTree**: A class that represents a min heap data structure which is a binary heap that makes sure to pertain a min heap property (the children elements are greater than their parent elements). The class performs methods like:
    - Insertion of a node.
    - Extract the minimum node.
    - Comparison of nodes.
    - Makes sure the Min Heap property remains through-out with the help of up_heapify and down_heapify methods.
    - Contains other simple functions that help in giving information of the parent node, left and right child, etc.

3. **Node**: A class that represents a node for a Red Black tree with attributes for the ride request, color, left and right child, parent node.

4. **RedBlackTree**: A class which represents a Red Black Tree data structure for managing ride requests based on their ride identifiers. The class contains methods to perform:
    - Initialization of an empty Red Black Tree.
    - Left rotation on a given node.
    - Right rotation on a given node.
    - Insertion of a ride request while maintaining the Red Black Tree properties.
    - Fixing the Red Black Tree after insertion.
    - Searching for a ride request based in the ride identifier.
    - Replacing one node with another in the Red Black Tree.
    - Finding the node with the minimum value in a subtree rooted at a given node.
    - Deletion of a ride request while maintaining the Red Black Tree properties.
    - Fixing the Red Black Tree properties after deletion.

5. **GatorTaxi**: A class which manages ride requests using a Min Heap Tree and a Red Black Tree, it performs the below mentioned functions:

- Initialization of a new GatorTaxi instance with a Min Heap Tree and a Red Black Tree.
- Insertion of a new ride request into both data structures after checking if it exists.
- Retrieval of the next ride with the lowest cost and shortest duration from the Min Heap Tree and deletion from the Red Black Tree.
- Cancellation of a ride request by deleting it from both data structures.
- Updating the trip duration and cost for a ride request, then re-inserting or canceling the ride accordingly.
- Printing the ride information for a specific ride request or a range of ride requests.
- A helper function that traverses the Red Black Tree in order to print the ride information for a variety of ride requests..

Function prototype:

1. **Class RideRequest:**
    - Init(self, rideIdentifier, estimatedCost, timeTaken)

2. **Class MinHeapTree:**
    - Init(self)
    - Parent_node(self, index)
    - Left_child_node(self, index)
    - Has_left_child(self, index)
    - Has_right_child(self, index)
    - Exchange_nodes(self, index_1, index_2)
    - Up_heapify(self, index)
    - Down_heapify(self, index)
    - Insertion(self, ride)
    - Extract_min_node(self)
    - Compare(self, ride_1, ride_2)

3. **Class Node:**
    - Init(self, color, left, right, parent_node)

4. **Class RedBlackTree:**
    - Init(self)
    - Left_rotation(self, x)
    - Right_rotation(self, x)
    - Insertion(self, ride)
    - Insertion_fix_up(self, z)
    - Search(self, rideIdentifer)
    - Replacing(self, u, v)
    - Minimum_value(self, x)
    - Deletion(self, z)

- Deletion_fixup(self,x)

5. **Class GatorTaxi:**
    - Init(self)
    - Insertion(self, ride)
    - getNextRide(self)
    - cancelRide(self, rideIdentifier)
    - updateTrip(self, rideIdentifer, new_timeTaken)
    - print(self, rideIdentifier_1, rideIdentifier_2)
    - printRange(self, node, rideIdentifier_1, rideIdentifier_2)

Time Complexity:

1. Insertion - MinHeap: O(log n)
   Red Black Tree: O(log n)
   Combined : O(log n)

2. GetNextRide - MinHeap: O(log n)
   Red Black Tree: O(log n)
   Combined : O(log n)

3. CancelRide - RedBlackTree: O(log n)
   MinHeap: O(n)
   Combined: O(n + log n)

4. UpdateTrip – RedBlackTree: O(log n)
   CancelRide: O(n + log n)
   Insert: O(log n)
   Combined: O(n + 2 * log n), where the CancelRide operation causes the most significant variable to be O(n).

5. Print – RedBlackTree: O(n)
   MinHeap: O(n)
   Combined: O(n)

Space Complexity:

O(n) for both Min Heap and Red Black Tree (n= no. of ride requests)