BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Work Integrated Learning Programmes Division
M.Tech (Data Science and Engineering)

(S1-18_DSEABZG519, S1-18_DSEADZG519, S1-18_DSEAHZG519) –
(Data Structures and Algorithms Design)
First Semester, 2018 -19

Assignment 1 – PS3 - [Traffic Fines] - [Weightage 10 %]

1. Problem Statement

In this Problem, you have to write an application in JAVA that keeps track of traffic fines.

All violations are noted by a traffic policeman in a file as a record <license number of driver, fine amount>. At the end of each day, files from all traffic policemen are collated. If a driver had been charged with more than three violations so far, then he has to be booked for further legal action. Also, the police department provides additional bonus to those policemen who have brought in large fine earnings. All policemen who have collected more than 90% of the maximum fine amount collected by any individual so far, shall be awarded the bonus.

1. Find out the drivers who are booked for legal action: All such license numbers are to be output in a file called "violators.txt".
2. Find out the policemen who are eligible for bonus: The list of policemen eligible for bonus must be output in a file called "bonus.txt".
3. **Perform an analysis of questions 1 and 2 and give the running time in terms of input size,n.**

Use hash tables for keeping track of drivers (and their violations), and a binary tree for keeping track of policemen (and their bookings).

Data structures to be used:

DriverHash: A separately chained hash table indexed by license numbers where each entry is of the form < license number, number of violations>. A simple hash function h(x) = x mod M, where M is the size of hash table can be used for this.

PoliceTree: This is a Binary Tree of entries of the form <police ID, total fine amount>

Basic PoliceTree node definition.

```
public class PoliceNode {
        int policeId, fineAmt;
        PoliceNode left, right;
        // include all basic functions for the node
}
```

Functions used by the application:

1. **void initializeHash(DriverHash dh):** This function creates an empty hash table that points to null.
2. **void insertHash(DriverHash dh, License lic):** This function inserts the license number lic to the hash table: if a driver's license number is already present, only the number of violations need to be updated else a new entry has to be created.
3. **void printViolators(DriverHash dh):** This function prints the serious violators by looking through all hash table entries and printing the license numbers of the drivers who have more than 3 violations onto the file "violators.txt".
4. **void destroyHash(DriverHash dh):** This function destroys all the entries inside hash table. This is a clean up code.
5. **PoliceTree insertByPoliceId(PoliceTree root, int policeId, int amount):** This function inserts an entry <id, amount> into the police tree ordered by police id. If the Police id is already found in the tree, then this function adds up to the existing amount to get the total amount to be stored against him. This function returns the updated tree.
6. **PoliceTree reorderByFineAmount(PoliceTree root):** This function reorders the Binary Tree on the basis of fine amount, instead of police id. This function removes the nodes from the original PoliceTree, and puts it in a new tree ordered by fine amount. Note that if the fine amount is equal to the amount in node j, then they will be inserted to the left of the node j. This function returns the root node of the new tree.
7. **void printBonusPolicemen(PoliceTree root):** This function prints the policemen who have earned more than 90% of largest fine amount collected by an individual on to a file called "bonus.txt".
8. **void destroyPoliceTree (PoliceTree root):** This function is a cleanup function that destroys all the nodes in the police tree.
9. **void printPoliceTree (PoliceTree root):** This function is meant for debugging purposes. This function prints the contents of the PoliceTree in-order.

Input File Sample

Every row of the input file should contain the <police id>,<license number>,<fine amount> in the same sequence. Save the input file as "input.txt"

Example:

111, 1231114, 100

111, 1214413, 200

222, 1231412, 100

222, 1231114, 100

2. Deliverables

a. PoliceNode.java file containing the basic node and associated functions

b. PoliceTree.java file containing the binary tree definition and all operation functions mentioned above.

   Note: For the purposes of testing, you may implement some functions to print the data structures or other test data.

   But all such functions must be commented before submission

c. DriverHash.java containing the hash table definition and associated functions

d. TrafficFines.java containing the public static void main function

e. Input.txt containing the input used to execute the program.

f. bonus.txt: The following entry will be present in each line of this file.<police id>

g. violators.txt: The following entry will be present in each line of this file.<license number>

h. A .txt file with answers of question no:3(Running times).

3. Instructions

- It is compulsory to make use of the data structure/s mentioned in the problem statement.
- It is compulsory to use JAVA for implementation.
- Ensure that all data structure insert and delete operations throw appropriate messages when their capacity is empty or full.
- For the purposes of testing, you may implement some functions to print the data structures or other test data. But all such functions must be commented before submission.
- Make sure that your read, understand, and follow all the instructions

4. Deadline

- The strict deadline for submission of the assignment is **Jan 5, 2019 eod.**
- Late submissions won't be evaluated.

5. How to submit

- This is a group assignment.
- Each group consists of 8-11 members from each batch. All members of the group will work on the same problem statement.

  Ex: DSAD_[BLR/HYD/DLH]_GROUP[1-5]

  The group details can be found here.
- Each group is further divided into subgroups.

  Ex: [BLR/HYD/DLH]_[B1-B4]_[G1-G5]_SUBGROUP [1]
- Each subgroup consists of 2-4 members.

- If your group has 11 students, there will be 3 subgroups of 3 students each and 1subgroup with 2 students. If your group has 10 students, there will be 2 subgroups of 3 students and 1 subgroup of 4 students.
- Each subgroup has to make one submission (only one, no resubmission) of solutions.
- Each sub group should zip the deliverables into "ASSIGNMENT1_[BLR/HYD/DLH]_[B1-B4]_[G1-G5]_SUBGROUP [1-4].zip" and upload in CANVAS in respective location under ASSIGNMENT Tab.
- Assignment submitted via means other than through CANVAS will not be graded

6. Evaluation

- The assignment carries 10 Marks
- Grading will depend on
    - Fully executable code with all functionality
    - Well-structured and commented code
- Every bug in the functionality will cost you 0.5 mark each.
- Every segmentation fault/memory violation will cost you 1 mark each.
- Source code files which contain compilation errors will get at most 25% of the value of that question.

7. Readings

- **Section 2.3,2.5,3.1:** Algorithms Design: Foundations, Analysis and Internet Examples Michael T. Goodrich, Roberto Tamassia, 2006, Wiley (Students Edition)