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 – PS1 - [Attendance Problem] - [Weightage 10 %]

1. Problem Statement

In an attendance monitoring system, the unique integer ID number of every employee is noted whenever the employee enters the organization. First time an employee enters into the office, the corresponding id is set to 1. From then onwards, each time an employee leaves the office premises for tea break or lunch break, and enters back this id is incremented. You can figure out that when this id is odd on a day, he is in the office premises and even when he is out.

The organization uses this data to perform the following analytics:

1. How many employees came today?
2. Did a particular employee come today?
3. How often did an employee enter into the office?
4. Which employee moves out of office most number of times?
5. Who all came within a range of IDs, and how often they entered?
6. **Perform an analysis for the questions above and give the running time in terms of input size,n.**

Implement the above problem statement in JAVA using BINARY TREE ADT, covered in the sessions.

The basic structure of the employee node will be:

```
public class EmployeeNode {

      int empId, attCount;
      EmployeeNode left, right;

      // include all basic functions for the node
}
```

Operations:

1. **EmployeeNode readEmployees(EmployeeNode emp, int id)**: This function reads the ids of employees entering the organization. One employee id can be populated per line (in the input text file) indicating their entry into the organization. The input data can be used to populate the tree. Use a trigger function to call this recursive function from the root node.
2. **int getHeadcount(EmployeeNode emp):** This function counts the number of unique IDs stored in the tree. Use a trigger function to call this recursive function from the root node.
3. **boolean searchID(EmployeeNode emp, int id):** This function searches whether a particular employees has entered the organization or not. The function returns true if the employees ID is found, else it returns false. Use a trigger function to call this recursive function from the root node.
4. **int howOften(EmployeeNode emp, int id):** This function returns how often the employee id entered the organization. The function returns the count of the number of times the employee entered the organization. Use a trigger function to call this recursive function from the root node.
5. **EmployeeNode frequentVisitor(EmployeeNode emp):** This function searches for the employee who has entered the most number of times and returns the node for that employee id. Use a trigger function to call this recursive function from the root node. For the sake of the assignment, you need to display any one of the employee ids if there are more than one employee who have entered the maximum number of times. For example, if employee id 22 and 23 have both visited 3 times, the output should show either 22 or 23.
6. **void printRangePresent(Employees emp, int id1, int id2):** This function prints the ids in the range **id1** to **id2** and how often they have entered the organization in a file name **output.txt**. For this purpose, the tree needs to be traversed; the id and frequency of the employees in the range must be printed. If the ID is found in the BT, its frequency cannot be zero as the person had entered the organization at least once.

   Note for entering inputs:

   You may enter all the integer employee ids given for input into a file named **input.txt**.

   Example:

   23

   22

   41

   121

   41

   22

   41

   121

If Input range is given as 23 to 125 for the employees within a range operation, the output should show:

23, 1

41, 3

121, 1

2. Deliverables

a. **EmployeeNode.java** file containing the basic node and associated functions

b. **EmpBT.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. **EmployeeAttendance.java** containing the public static void main function, operation menu and switch cases with input requests where relevant.

d. Word document showing a successful run and corresponding output for each of the operations.

e. A .txt file with answers of question no:6(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:** Algorithms Design: Foundations, Analysis and Internet Examples Michael T. Goodrich, Roberto Tamassia, 2006, Wiley (Students Edition)