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 – PS4 - [StudentRecord] - [Weightage 10 %]

1. Problem Statement

In this Problem, you have to write an application in JAVA that keeps track of student records in a university.

At a university, there is a need to store all the details of graduating students. For this exercise, let us consider that the CGPA of the student is stored against the student id.

The students ID has the following format <YYYYAAADDDD> where

YYYY - represents the year in which this student joined the university

AAA - a three letter (alphabet) representing degree program

DDDD - a four digit number representing the students roll number

For instance, an ID can be of the form 2008CSE1223 corresponding to a student joined the university in the year 2008 in the CSE department with the roll number 1223. The university offers a 4 year graduate degree program in CSE (Computer Science and Engineering), MEC (Mechanical Engineering), ECE (Electronics and Communication Engineering) and ARC (Architecture). In the year 2008 the first batch of 20 students were admitted to the university. Now in the year 2018, 200 students were admitted across all departments. Create an input file **input.txt** with a random list of students per year and their corresponding CGPA.

The university now wants to use the details of all its past students to

1. Identify and commemorate their alumni on the 10^{th} year anniversary of the University. For this they will need to get a list of all students who scored over x CGPA.
2. Extend a new course offering to selected students who have graduated in the past five years. For this they will need to get a list of all students who secured between CGPA x to CGPA y in the past five years.
3. Identify the maximum and average CGPA per department.

- **Design a hash table, which uses student Id as the key to hash elements into the hash table**. Generate necessary hash table definitions needed and provide a design document (2 page) detailing clearly the design and the details of considerations while making this design.

- Design a hash function **HashId()** which accepts the student-ID as a parameter and returns the hash value. You are only allowed to use basic arithmetic and logic operations in implementing this hash function. Write as a comment to this function, the reasons for the specific choice of hash function.

- Create / populate the hash table from the list of student ID and the corresponding CGPA given in the input file.

Functions used by the program:

1. **void initializeHash(StudentHash records):** This function creates an empty hash table and points to null.
2. **void insertStudentRec(StudentHash records, String studentId, float CGPA):** This function inserts the student id and corresponding CPGA into the hash table. The file read can happen outside the function and only the information in every individual row needs to be passed to the function.
3. **void hallOfFame(StudentHash records, float CGPA):** This function prints the list of all students who have a CGPA greater than the CGPA passed to the function. This hall of fame list should be output in a file **halloffame.txt** and should contain the studentid and CGPA.
4. **void newCourseList(StudentHash records, float CGPAFrom, float CPGATo):** This function prints the list of all students who have a CGPA within the given range and have graduated in the last 5 years. This list should be output to a file **courseOffer.txt** that contains the student id, CGPA and year of graduation.
5. **void depAvg(StudentHash records):** This function prints the list of all departments followed by the maximum CGPA and average CGPA of all students in that department. The output should be captured in a file **departmentAverage.txt** that contains this data in the following format, <departmentID>, Max: <max CGPA> , Avg: <avg CGPA>
6. **void destroyHash(StudentHash records):** This function destroys all the entries inside hash table. This is a clean up code.
7. Include all other functions that are required to support these basic mandatory functions.

Input File Sample

Every row of the input file should contain the <student id> / <CGPA> separated by a fixed delimiter. Save the input file as “input.txt”

Example:

2008CSE1223 / 3.5

2008CSE1224 / 3.9

2008CSE1225 / 4.5

2008CSE1226 / 4.8

2008CSE1227 / 4.1

2008MEC1001 / 4.2

2. Deliverables

1. StudentHash.java containing the hash table definition and associated functions
2. StudentRecords.java containing the public static void main function
3. Input.txt containing the input used to execute the program.
4. halloffame.txt containing the list of hall of fame students and their CGPA
5. courseOffer.txt containing the list of students who are eligible for the new course offering
6. departmentAverage.txt containing the list of departments and the max and average CGPA.

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