(S2-18_DSECFZG519)
(Data Structures and Algorithms Design)
Academic Year 2018-2019

Assignment 1 – PS4 - [Student Record] - [Weightage 13%]

1. Problem Statement

In this Problem, you have to write an application in Python 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 who 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 (maximum of 5.0 point CGPA).

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

a. Identify and commemorate their alumni on the 10th year anniversary of the University. For this they will need to get a list of all students who scored over x CGPA.
b. Extend a new course offering to selected students who have secured a CGPA between a specified range. 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.
c. 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 (1 page) detailing clearly the design and the details of considerations while making this design and the reasons for the specific choice of hash function.

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.

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

Functions:

1. **def initializeHash(self):** This function creates an empty hash table and points to null.
2. **def insertStudentRec(StudentHashRecords, studentId, CGPA):** This function inserts the student id and corresponding CPGA into the hash table. The inputs need to be read from a file **inputPS4.txt** which contains the student ID and overall graduating CGPA (decimal value with a maximum of 5.0 points). The file read can happen outside the function and only the information in every individual row needs to be passed to the function.
3. **def hallOfFame(StudentHashRecords, CGPA):** This function prints the list of all students who have a CGPA greater than the CGPA passed to the function. The input CGPA can be read from the file **promptsPS4.txt**. The input can be identified with the tag mentioned below

   hallOfFame: 4.5

   This hall of fame list should be output in a file **outputPS4.txt** and should contain the studentid and CGPA.
   Output format:
   ---------- hall of fame ----------
   Input: 4.5
   Total eligible students: 2
   Qualified students:
   2008CSE1225 / 4.5
   2008CSE1226 / 4.8
   ---

4. **def newCourseList(StudentHashRecords, CGPAFrom, 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. The input CGPAs can be read from the file **promptsPS4.txt**. The input can be identified with the tag mentioned below

   courseOffer: 3.5 : 4:0

   This list should be output to **outputPS4.txt** that contains the student id and CGPA.
   Output format:
   ---------- new course candidates ----------
   Input: 3.5 to 4.0

Total eligible students: 2
Qualified students:
2008CSE1223 / 3.5
2008CSE1224 / 3.9
---

5. **def depAvg(StudentHashRecords):** 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 **outputps4.txt** following format
   Output format:
   ---------- department CGPA ----------
   CSE: max: 3.5, avg: 3.4
   MEC: max: 3.5, avg: 3.4
   ECE: max: 3.5, avg: 3.4
   ARC: max: 3.5, avg: 3.4

   (Note: the above output numbers are indicative and not actuals.)

6. **def destroyHash(StudentHashRecords):** 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.

2. Sample file formats

Sample Input file

Every row of the input file should contain the <student id> / <CGPA> separated by a slash (/). The year of the student ids in the file cannot be greater than 2014 (considering that the file contains the CGPA of only graduating students). Consider a CPGA scale of 0.0 to 5.0. Save the input file as **inputPS4.txt**

Sample inputPS4.txt

2008CSE1223 / 3.5
2008CSE1224 / 3.9
2008CSE1225 / 4.5
2008CSE1226 / 4.8
2008CSE1227 / 4.1
2008MEC1001 / 4.2

Sample promptsPS4.txt

hallOfFame: 4.5
courseOffer: 3.5 : 4:0

Sample outputPS4.txt

---------- hall of fame ----------
Input: 4.5
Total eligible students: 2
Qualified students:
2008CSE1225 / 4.5
2008CSE1226 / 4.8

---------- new course candidates ----------
Input: 3.5 to 4.0
Total eligible students: 2
Qualified students:
2008CSE1223 / 3.5
2008CSE1224 / 3.9

---------- department CGPA ----------
CSE: max: 3.5, avg: 3.4
MEC: max: 3.5, avg: 3.4
ECE: max: 3.5, avg: 3.4
ARC: max: 3.5, avg: 3.4
----

3. Deliverables

a. **Zipped PS4_SR_[Group id].py package folder** containing modules and package files for the entire program code and associated functions
b. **inputPS4.txt** file used for testing
c. **promptsPS4.txt** file used for testing
d. **outputPS4.txt** file containing the output of the program
e. **analysisPS4.txt** file containing the running time analysis for the program.

4. Instructions

a. It is compulsory to make use of the data structure/s mentioned in the problem statement.
b. It is compulsory to use Python for implementation.
c. Ensure that all data structure insert and delete operations throw appropriate messages when their capacity is empty or full.
d. 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.
e. Make sure that your read, understand, and follow all the instructions

f. Ensure that the input, prompt and output file guidelines are adhered to. Deviations from the mentioned formats will not be entertained.

g. Run time analysis is provided in asymptotic notations and not timestamp based runtimes in sec or milliseconds.

5. Deadline

a. The strict deadline for submission of the assignment is 6th July, 2019.

b. The deadline is set for a month from the date of rollout to accommodate for the semester exams. No further extension of the deadline will be entertained.

c. Late submissions will not be evaluated.

6. How to submit

a. This is a group assignment.

b. Each group has to make one submission (only one, no resubmission) of solutions.

c. Each group should zip the deliverables and name the zipped file as below

d. “ASSIGNMENT1_[BLR/HYD/DLH/PUN/CHE]_[B1/B2/…]_[G1/G2/…].zip”

e. and upload in CANVAS in respective location under ASSIGNMENT Tab.

f. Assignment submitted via means other than through CANVAS will not be graded.

7. Evaluation

a. The assignment carries 13 Marks.

b. Grading will depend on

   a. Fully executable code with all functionality

   b. Well-structured and commented code

   c. Accuracy of the run time analysis

c. Every bug in the functionality will have negative marking.

d. Source code files which contain compilation errors will get at most 25% of the value of that question.

8. Readings

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