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

(S1-20_DSECFZG519)

(Data Structures and Algorithms Design)

Academic Year 2020-2021

Assignment 1 – PS26 - [Club House] - [Weightage 12%]

1. Problem Statement

In this Problem, you have to write an application in Python 3.7 that keeps track of club members and their details.

A newly opened club house is inviting guests to register into their membership program. They have received a lot of entries and need help retrieving details of their applicants quickly. In order to do this, they need your help in designing a system that can quickly save and find the applicant details based on the applicant's name. The details that need to be included are:

1. Applicant Name
2. Phone Number
3. Member reference
4. Application status

The club house would like to use the system to provide the following functionality

1. Add names of applicants and other details into the system
2. Find and update applicant details based on applicant's name
3. Generate a list of applicants who have been referred by a particular member
4. Generate a report on the number of applications in the various stages of processing (Applied, Verified, Approved)

Design a hash table, which uses applicant's name 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 applicant's name as a parameter and returns the hash value.

Create / populate the hash table from the list of applicant names and the corresponding details given in the input file.

Operations:

1. **def initializeHash(self):** This function creates an empty hash table and points to null.
2. **def insertAppDetails(ApplicationRecords, name, phone, memRef, status):** This function inserts the applicant's name and corresponding details into the hash table. The inputs need to be read from a file **inputPS26.txt** which contains the all the applicant details. The file read can happen outside the function and only the information in every individual row needs to be passed to the function. Each applicant's details should be recorded in one row separated by a slash ("/") as shown below.

   Sample inputPS26.txt file entries

   Aravind Shetty / 9988112311 / 11321 / Applied

   Deepak Prasad / 9923212234 / / Applied

   Sandhya Raman / 9213231311 / 11129 / Verified

   Joginder Singh / 8234219326 / 21299 / Applied

   Vinay Shah / 9912356788 / 11129 / Approved

   …..

   After successfully inserting the applicant details, a summary of the insert status should be output to the file **outputPS26.txt** in the below format.

   Successfully inserted xx applications into the system.

   Where xx is the number of applications.

3. **def updateAppDetails(ApplicationRecords, name, phone, memRef, status):** This function finds the applicant's details based on the name and updates the corresponding details into the hash table. The inputs need to be read from a file **promptsPS26.txt** which contains the below tag to indicate an update.

   Update: Deepak Prasad / 9923212234 / / Verified

   After the update is done, the update summary should be sent to the outputPS26.txt file in the below format.

   Updated details of xxxxxxxxxx. Yyyy has been changed.

Where xxxx is the Name of the applicant and yyyy is the field(s) that has/have been updated.

4. **def memRef(ApplicationRecords, memID):** This function prints the list of all applicants who have been referred by a particular member. The member number can be read from the file **promptsPS26.txt**. The input can be identified with the tag mentioned below

   memberRef: 11129

   The list of applicants should be output in a file **outputPS26.txt** and should contain the applicant's details including name, phone number and application status.

   Output format:

   ---------- Member reference by 11129 ----------

   Sandhya Raman / 9213231311 / Verified

   Vinay Shah / 9912356788 / Approved

   ---

5. **def appStatus(ApplicationRecords):** This function prints the list of number of applications in each stage of the application process including Applied, Verified and Approved. This function is triggered when the below tag is encountered in the **promptsPS26.txt** file. The input can be identified with the tag mentioned below

   appStatus

   This list should be output to **outputPS26.txt** that contains the number of applications in each status.

   Output format:

   ---------- Application Status ----------

   Applied: 3

   Verified: 1

   Approved: 1

   ---

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

Sample file formats

Sample Input file

Each row of the input file should correspond to one application. Each row of the input file will contain the applicant's name, phone number, member reference id and application status. Each detail will be separated by a slash as shown below. Save the input file as **inputPS26.txt**

Sample inputPS26.txt

Aravind Shetty / 9988112311 / 11321 / Applied

Deepak Prasad / 9923212234 / / Applied

Sandhya Raman / 9213231311 / 11129 / Verified

Joginder Singh / 8234219326 / 21299 / Applied

Vinay Shah / 9912356788 / 11129 / Approved

Sample promptsPS26.txt

Update: Deepak Prasad / 9923212234 / / Verified

memberRef: 11129

appStatus

Sample outputPS26.txt

Successfully inserted 5 applications into the system.

Updated details of Deepak Prasad. Application Status has been changed.

---------- Member reference by 11129 ----------

Sandhya Raman / 9213231311 / Verified

Vinay Shah / 9912356788 / Approved

---------- Application Status ----------

Applied: 3

Verified: 1

Approved: 1

Note that the input/output data shown here is only for understanding and testing, the actual file used for evaluation will be different.

2. Deliverables

1. Word document **designPS26_<group id>.docx** detailing your design and time complexity of the algorithm.
2. **[Group id]_Contribution.xlsx** mentioning the contribution of each student in terms of percentage of work done. Download the Contribution.xlsx template from the link shared in the Assignment Announcement.
3. **inputPS26.txt** file used for testing
4. **promptsPS26.txt** file used for testing
5. **outputPS26.txt** file generated while testing
6. **.py file** containing the python code. Create a single *.py file for code. Do not fragment your code into multiple files

Zip all of the above files including the design document and contribution file in a folder with the name:

[Group id]_A1_PS26_Club.zip and submit the zipped file.

Group Id should be given as **Gxxx** where xxx is your group number. For example, if your group is 26, then you will enter G026 as your group id.

3. Instructions

1. It is compulsory to make use of the data structure(s) / algorithms mentioned in the problem statement.
2. Ensure that all data structure insert and delete operations throw appropriate messages when their capacity is empty or full. Also ensure basic error handling is implemented.
3. 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.
4. Make sure that your read, understand, and follow all the instructions
5. Ensure that the input, prompt and output file guidelines are adhered to. Deviations from the mentioned formats will not be entertained.
6. The input, prompt and output samples shown here are only a representation of the syntax to be used. Actual files used to evaluate the submissions will be different. Hence, do not hard code any values into the code.
7. Run time analysis is to be provided in asymptotic notations and not timestamp based runtimes in sec or milliseconds.

Instructions for use of Python:

1. Implement the above problem statement using Python 3.7.
2. Use only native data types like lists and tuples in Python, do not use dictionaries provided in Python. Use of external libraries like graph, numpy, pandas library etc. is not allowed. The purpose of the assignment is for you to learn how these data structures are constructed and how they work internally.
3. Create a single *.py file for code. Do not fragment your code into multiple files.
4. Do not submit a Jupyter Notebook (no *.ipynb). These submissions will not be evaluated.
5. Read the input file and create the output file in the root folder itself along with your .py file. Do not create separate folders for input and output files.

4. Deadline

1. The strict deadline for submission of the assignment is **27th Dec, 2020.**
2. The deadline has been set considering extra days from the regular duration in order to accommodate any challenges you might face. No further extensions will be entertained.
3. Late submissions will not be evaluated.

5. How to submit

1. This is a group assignment.
2. Each group has to make one submission (only one, no resubmission) of solutions.
3. Each group should zip all the deliverables in one zip file and name the zipped file as mentioned above.
4. Assignments should be submitted via Canvas > Assignment section. Assignment submitted via other means like email etc. will not be graded.

6. Evaluation

1. The assignment carries 12 Marks.
2. Grading will depend on
    a. Fully executable code with all functionality working as expected
    b. Well-structured and commented code
    c. Accuracy of the run time analysis and design document.
3. Every bug in the functionality will have negative marking.
4. Marks will be deducted if your program fails to read the input file used for evaluation due to change / deviation from the required syntax.
5. Use of only native data types and avoiding libraries like numpy, graph and pandas will get additional marks.

6. Plagiarism will not be tolerated. If two different groups submit the same code, both teams will get zero marks.
7. Source code files which contain compilation errors will get at most 25% of the value of that question.

7. Readings

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