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

(Data Structures and Algorithms Design)

Academic Year 2022-2023

Assignment – PS07- [Weightage 25%]

1. Problem Statement –Question 1

The graduate school of a University requires your help in developing a new system to save details of students who apply for their various master's degree programs. They have received a lot of entries and need help retrieving details of their applicants quickly. To do this, they need your help in designing a system that can quickly save and find the student details based on their name. The details that need to be included are:

a) Applicant Name
b) Phone Number
c) Resident Country
d) Program Applied
e) Application status

The department would like to use the system to provide the following functionality

1. Add applicant names and other details into the system
2. Find and update applicant details based on their name
3. Generate a list of applicants who have been accepted for a particular program
4. Generate a report on the number of applications in the various stages of processing (Applied, Rejected, Approved)

Design a hash table, which uses student'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) Clearly detailing 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.

Requirements:

1. Implement the above problem statement using Python 3.7
2. Read the input from a file(**inputPS07Q1.txt**), where each applicant's details should be recorded in one row separated by a slash ("/")
3. You will output your answers to a file (**outputPS07Q1.txt**) for each line.
4. Perform an analysis for the features above and give the running time in terms of input size: n.

Operations:

1. **def initializeHash(self):** This function creates an empty hash table and points to null.
2. **def insertAppDetails(ApplicationRecords, name, phone, country, program, status):** This function inserts the student's name and corresponding details into the hash table. The inputs need to be read from a file **inputPS07.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 inputPS07Q1.txt file entries

   ```
   aaa / 9977112388 / India / Computer Science/ Applied
   bbb / 9953152234 /Nepal / Electronics / Applied
   ccc / 9244231355 / India / Computer Science / Applied
   ddd / 9287516571 / Bangladesh / Biotechnology / Applied
   eee / 9982154788 / India / Computer Science / Approved
   …..
   ```

   After successfully inserting the applicant details, a summary of the insert status should be output to the file **outputPS07Q1.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, country,**

program, 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 **promptsPS07Q1.txt** which contains the below tag to indicate an update.

Update: ccc / 9244231355 / India / Computer Science / Rejected

After the update is done, the update summary should be sent to the **outputPS07Q1.txt** file in the below format.

Updated details of xxxx. 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, Program): This function prints the list of all applicants who have applied to a particular program. The program name can be read from the file **promptsPS08Q1.txt**. The input can be identified with the tag mentioned below

Program: Computer Science

The list of applicants should be output in a file **outputPS07Q1.txt** and should contain the applicant's details including name, phone number, country, and application status.

Output format:

```
---------- Applicants for Computer Science ----------
aaa / 9977112388 / India / Computer Science/ Applied
ccc / 9244231355 / India / Computer Science / Rejected
eee / 9982154788 / India / Computer Science / Approved
```

5. def appStatus(ApplicationRecords): This function prints the list of number of applications in their current stage of the application process including Applied, Rejected and Approved. This function is triggered when the below tag is encountered in the **promptsPS07Q1.txt** file. The input can be identified with the tag mentioned below

appStatus

This list should be output to **outputPS07Q1.txt** that contains the number of applications in each status.

Output format:

```
---------- Application Status ----------
```

```
Applied: 3
Rejected: 1
Approved: 1
-------------------------------------
```

6. def destroyHash(ApplicationRecords): This function destroys all the entries

I nside hash table. This is a clean-up code.

7. Include all other functions that are required to support these basic mandatory functions.

Sample Input:

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, country, program, and application status. Each detail will be separated by a slash as shown below. Save the input file as **inputPS07Q1.txt**

Sample inputPS07Q1.txt

```
aaa / 9977112388 / India / Computer Science/ Applied
bbb / 9953152234 /Nepal / Electronics / Applied
ccc / 9244231355 / India / Computer Science / Applied
ddd / 9287516571 / Bangladesh / Biotechnology / Applied
eee / 9982154788 / India / Computer Science / Approved
```

Sample promptsPS07Q1.txt

```
Update: ccc / 9244231355 / India / Computer Science / Rejected

Program: Computer Science

appStatus
```

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

Sample Output:

Display the output in **outputPS07Q1.txt**.

Sample outputPS07Q1.txt

```
Successfully inserted 5 applications into the system.

Updated details of ccc. Application Status has been changed.

---------- Applicants for Computer Science ----------
aaa / 9977112388 / India / Computer Science/ Applied
ccc / 9244231355 / India / Computer Science / Rejected
eee / 9982154788 / India / Computer Science / Approved

-------------------------------------

---------- Application Status ----------
```

Applied: 3

Rejected: 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 **designPS07Q1_<student_id>.docx** detailing your design and time complexity of the algorithm.
2. **InputPS07Q1.txt and PromptsPS07Q1.txt** files are used for testing
3. **OutputPS07Q1.txt** file generated while testing
4. **.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 file in a folder with the name:

   [Student id]_PS07Q1.zip .

3. Problem Statement –Question 2

You are the leader of an elite task force that is commissioned to guard the borders of your country. You are allowed to build a team of your choice and each member can be equipped with one unique weapon. Each weapon fires a different type of ammunition and has a different capability of damage. The ammunition for these weapons come in packs of 100 units and have specific weights. Collectively, the team can carry a total weight of x kgs. As the leader of the task force it is your responsibility to decide the ratios of ammunition (full packs or partial packs) the team caries in order to maximize damage capability.

Requirements:

1. Formulate an efficient algorithm using Greedy Method to determine which ammunition to carry in what ratio to maximize damage capability.
2. Analyse the time complexity of your algorithm.
3. Implement the above problem statement using Python 3.7.

Sample Input:

For example, if there are 6 different team members carrying 6 weapons with a total ammunition carrying capacity of 118 kgs. Find the ratios in which each ammunition should be taken such as to maximize damage

Ammunition (i)	Weight per pack	Damage per pack
1	10	20
2	30	40
3	18	38
4	80	60
5	10	15
6	20	22

Input should be taken in through a file called “**inputPS07Q2.txt**” which has the fixed format mentioned below using the “/” as a field separator:
Weapons : <count>
MaxWeight : <weight>
<Ammunition i> / <Weight i> / <Damage i>

Ex:

```
Weapons : 6
MaxWeight : 118
A1 / 10 / 20
A2 / 30 / 40
A3 / 18 / 38
…
```

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

Sample Output:

```
Total Damage: 157.2
Ammunition Packs Selection Ratio:
A1 > 1
A2 > 1
A3 > 1
A4 > 0.37
A5 > 1
A6 > 1
```

Note that the input/output data shown here is only for understanding and testing, the actual file used for evaluation will be different.
Display the output in **outputPS07Q2.txt**.

4. Deliverables

1. Word document **designPS07Q2_<student id>.docx** detailing your design and time complexity of the algorithm.
2. **inputPS07Q2.txt** file used for testing
3. **outputPS07Q2.txt** file generated while testing
4. **.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 file in a folder with the name:

[Student id]_ PS07Q2.zip

5. 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.

6. Deadline

1. The strict deadline for submission of the assignment is **<Refer E-learn Course page>.**
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.

7. How to submit

1. All the deliverables(zip files from section 2 and 5) must be combined in one zip file and named as **<Student ID>.zip**
2. Assignments should be submitted via E-Learn > Assignment section. Assignment submitted via other means like email etc. will not be graded.

8. Evaluation

1. The assignment carries **13 +12 =25 Marks**.
2. Grading will depend on
    i. Fully executable code with all functionality working as expected
    ii. Well-structured and commented code
    iii. 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.

9. Readings

Text book: Algorithms Design: Foundations, Analysis and Internet Examples Michael T. Goodrich, Roberto Tamassia, 2006, WileyGoodrich, Roberto Tamassia, 2006, Wiley (Students Edition). **Chapters:** 2.5,5.1