1. Problem Statement

At the government hospital treating patients for Covid, the management is preparing a token system that gives priority to senior citizens who want to get tested over other patients. As the patients keep coming and registering, they are added to a priority queue from which patient's names are called out for testing. Anticipating a rise in the number of cases across the world, an appointment software has to be developed to manage this priority effectively.

The application should be able to:

1. Take the patient's age and create a patient ID in the following format: <xxxxyy> where

   xxxx is the patient id and

   yy is the patient's age

2. Insert the patient id in the priority queue based on the age of the patient.
3. If x testing counters got vacant, display the next set of patient IDs and corresponding names that should go for testing and remove them from the priority queue.

Requirements:

1. **Implement the above problem statement in Python 3.7 using Heaps.**

2. **Perform an analysis for the questions above and give the running time in terms of input size: n.**

Data Structures to be used:

PatientRecord: A list containing the patient information including the patient's name, age and the patient number (assigned by the program).

TestingQueue: A max heap containing the patient id sorted in order of next patient for testing based on the age of the patient.

Patient Record data structure:

```python
class PatientRecord:
    def __init__(self, age, name, Pid):
        self.PatId = str(Pid) + str(age)
        self.name = name
        self.age = age
        self.left = None
        self.right = None
```

Operations:

1. **def registerPatient(self, name, age):** This function registers the name and age of the patient entering the hospital and assigns them an ID that is then used to capture the details of the patient in the Patient Record. When the program is executed for the first time, the patient details are loaded from an input file **inputPS1a.txt**. This is analogous to the list of patients present at the hospital before the hospital opens.

   InputPS1a format:

   Surya, 60

   Ajay, 54

   Rishi, 57

   After all records are read from the inputPS1a file, and the queue is sorted, the queue should be output to the file outputPS1.txt in the below format.

   ---- registered Patient ---------------

   No of patients added: 3

   Refreshed queue:

   100160, Surya

   100357, Rishi

   100254, Ajay

   Thereafter, new patients will be input through another file **inputPS1b.txt** and identified with the tag **newPatient.**

   newPatient: John, 55

   After every new patient in the inputPS1b is inserted into the patient record the heap will have to be refreshed and the new queue should be output to the file **outputPS1.txt** in the below format.

2. **def enqueuePatient(self, PatId)**: This function assigns the patient a place in the max heap depending on their age. The patient id is inserted into the max heap. This function should be called every time a new patient is added and should run a sort after adding the patient id to keep the testing queue updated as per the age condition.

3. **def nextPatient(self):** This function prints the next x number of patient_IDs and their names that are next in line for testing. This function is called when the program encounters a next patient tag from the **inputPS1b.txt** file. The format of the next patient will be

   nextPatient: 1

   The function will read the x number of patient details that should be output. The output is pushed to the file **outputPS1.txt** in the below format.

   ---- next patient : 1 ---------------

   Next patient for testing is: 100260, Surya

   ----

4. **def _dequeuePatient(self, PatId):** This function removes from the queue the patient ID that has completed the testing and updates the queue. The function is called from the nextPatient function itself after the next patients name is displayed.

5. Include all other functions required to support the operations of these basic functions.

Sample file formats

   Sample Input file

   The inputPS1.txt file contains the first set of registrations.

Sample inputPS1a.txt

Surya, 60

Ajay, 54

Rishi, 57

Sample inputPS1b.txt

newPatient: John, 55

nextPatient: 1

newPatient: Pradeep, 45

nextPatient: 2

nextPatient: 1

newPatient: Sandeep, 60

nextPatient: 3

Sample outputPS1.txt

---- registered Patient ---------------

No of patients added: 3

Refreshed queue:

100160, Surya

100357, Rishi

100254, Ajay

---- new patient entered---------------

Patient details: John, 55, 100455

Refreshed queue:

100260, Surya

100357, Rishi

100455, John

100554, Ajay

---- next patient: 1 ---------------

Next patient for testing is: 100260, Surya

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 **designPS1_<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. **inputPS1a.txt** and **inputPS1b.txt** file used for testing

4. **outputPS1.txt** file generated while testing

5. **.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_PS1_Testing.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 **10th April, 2022.**
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 E-Learn > 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:** 2.4