There are 1 questions for a total of 100 points.

---

(100) 1. We would like to develop some software for Rendezvouz-2017 which will help you keep track of scores in various events. We would like to maintain data of participants and their scores in various events. Assume that there is some standardised scoring mechanism for different events so that we can compare the performance of participants across different events (e.g., if A gets a score 7 in event E_1 and B gets a score 8 in event E_2, then B has better performance than A). For each individual event, we would like to access the best participants in that event. This can be done by maintaining a max heap for the event where the keys are the scores of the participants in that event. We would also like to maintain a data structure for accessing the best participants across all the events. For this we maintain a heap of heaps.

Each event has a unique event ID. The other information attached with every event is the name and description of the event. So, it makes sense to define and use a class `Event`. You may assume that there will be at most 1000 events in Rendezvouz-2017.

```
Event{
   String eventID;
   String eventName;
   String eventDescription;
}
```

Each participant of Rendezvouz-2017 gets a unique paricipant ID. Attached with each participant is the name of the participant and the name of his/her university. So, `Participant` class below may be useful. Feel free to add methods in this class if needed. You may assume that the number of participants is at most 10^6 (this year's Rendezvouz may be really popular... our software should be prepared!)

```
Participant{
   String participantID;
   String participantName;
   String universityName;
}
```

For each event, you should maintain a max heap where the value of each node is a participant and key is the score of the participant in the event. We would like to maintain a max-heap because we would like to access top performers in the event and we know that max-heap is a fast data structure supporting such operations.

We would also like access overall top performers across all events. For this, you should maintain a Heap of Event Heaps which is explained next.

Heap of Heaps: For each individual event, we maintain a max heap where the key is the standardised score and the value is the details about the participant (i.e., object of class Participant). We maintain another max heap where each node corresponds to an event. For each node, the key is the best score of a participant in that event and the value is the max heap for that event. Maintaining such a heap of event heaps, allows us access top performers across all events.

Task: You should do an appropriate implementation of event heap and heap of heaps. You should also write a program called `Simulate.java` that reads instructions/queries from a file named `query.txt` that has one instruction per line. The different kind of queries/instructions are discussed next.

- Add a participant: This adds a new participant into the system. Example of this instruction is given below:
  `ADD PARTICIPANT P00001, Ashwin, IIT Mumbai`
  (P00001 is the participant ID of Ashwin who is from IIT Mumbai)
- Add an event: This is an instruction for adding a new event into the system. Here is an example:
  `ADD EVENT E0001, Quiz, Quiz competition`
  (E0001 is the event ID of the Quiz event)
- Add a participant to an event: This is self explanatory. Here is the example:
  `ADD P00001, E0001`
  (This means that Ashwin should be added to the quiz event.)
- Update score of a participant in an event: This is self-explanatory. Here is an example:
  `UPDATE SCORE P00001, E0001, 7`
  (This means that the score of Ashwin in the Quiz event should be updated to 7. You may assume that the default score of a participant is 0.)
- Delete a participant in an event: Sometimes a participants is not able to make it to an event or a participant gets disqualified. In such cases, we would like to delete this participant from this event. Here is an example of this instruction:
  `DELETE EVENT PARTICIPANT P00001, E0001`
  (This means that Ashwin should be removed from the Quiz event.)
- Delete Participant: Sometimes a participant is not able to make it to Rendezvouz or gets disqualified from all events due to bad behaviour. In that case, we would like to delete this participant from all events. Here is an example:
  `DELETE PARTICIPANT P00001`
  (This means that Ashwin should be deleted from all events.)
- Cancel Event: Sometimes some pre-planned event gets cancelled due to organising issues. An event may also get cancelled after it has been held because of complaints of unfairness. In such cases, we would like to delete an event. Here is an example:
  `DELETE EVENT E0001`
  (This means that the quiz event should be deleted.)
- Top 3 in an event: This instruction requests to output the top 3 performers (in that order) in a given event along with the scores. The usual rules for top 3 performers should apply. For example, consider four participants in an event `E0002` – (P00002, A, U1, 8), (P00003, B, U1, 8), (P00004, C, U2, 7), (P00005, D, U3, 6). That is participant A with participant id P00002 from U1 university has a score 8 in this event and so on. Then on instruction:
  `TOP3 IN EVENT E0002`
  The output (on standard output) should be:
  ```
  P00002, A, U1, 8
  P00003, B, U1, 8
  P00004, C, U2, 7
  ```
- Top 3: This is similar to the above but the output should be across all events and the output should also contain the information about the event. For example, suppose the overall top performers are (P00006, X, U1, E0002, E1, 8), (P00007, Y, U1, E0001, E2, 8), (P00008, Z, U6, E0005, E1, 7). This is, participant X with participant id P00006 and university U1 received 8 points in event E1 with event id E0002 and so on. Then on instruction:
  `TOP3`
  The output (on standard output) should be:
  `P00006, X, U1, E0002, E1, 8`

```
    P00007, Y, U1, E0001, E2, 8
    P00008, Z, U6, E0005, E1, 7
```

Here is an example file `Query.txt`:

```
ADD EVENT E1, Jeopardy, Quizzing
ADD EVENT E2, Sports Quiz, Quizzing
ADD EVENT E3, Kombat, Quizzing
ADD PARTICIPANT P1, Vijay, BCCI
ADD PARTICIPANT P2, Virat, BCCI
ADD PARTICIPANT P3, Ashwin, BCCI
ADD PARTICIPANT P4, Pujara, BCCI
ADD PARTICIPANT P5, Jadeja, BCCI
ADD PARTICIPANT P6, Rahul, BCCI
ADD P1, E1
ADD P3, E1
ADD P4, E1
ADD P5, E1
ADD P1, E2
ADD P2, E2
ADD P5, E2
ADD P6, E2
ADD P1, E3
ADD P3, E3
ADD P4, E3
ADD P5, E3
UPDATE SCORE P5, E1, 6
UPDATE SCORE P6, E2, 2
UPDATE SCORE P5, E3, 6
UPDATE SCORE P3, E1, 5
UPDATE SCORE P3, E3, 8
UPDATE SCORE P2, E2, 9
UPDATE SCORE P4, E1, 7
UPDATE SCORE P5, E2, 6
UPDATE SCORE P4, E3, 3
UPDATE SCORE P1, E1, 9
UPDATE SCORE P1, E2, 5
UPDATE SCORE P1, E3, 7
TOP3 IN EVENT E2
DELETE PARTICIPANT P1
TOP3
```

For the above the query file the output (on standard output) should be the following:

```
P2, Virat, BCCI, 9
P5, Jadeja, BCCI, 6
P1, Vijay, BCCI, 5
P2, Virat, BCCI, E2, Sports Quiz, 9
P3, Ashwin, BCCI, E3, Kombat, 8
P4, Jadeja, BCCI, E1, Jeopardy, 7
```

For any illegal query (for example adding a participant to an event that has not been added yet) your software should throw an exception.

Evaluation: Evaluation of homework consists of the following two components:

1. Submission component (75 points): Your code will be checked for correctness and running time. This will partially be done using an automated scripts. So, please make sure that you strictly follow these instructions:
   - Write all your code in a directory named ⟨Your entry number⟩.
   - For submission, create a zip file named ⟨Your entry number⟩.zip of your directory and submit. Details of where to submit will be sent in some time.
2. Viva component (25 points): We will hold all the labs during an evaluation week and you will be expected to attend the lab. During the lab, the following will be done:
   - You will be asked to make a small addition in your program to check your understanding.