There are 1 questions for a total of 100 points.

---

(100) 1. We would like to develop some software for Training and Placement (T&P cell) at IIT Delhi for helping them to match IIT Delhi graduates to appropriate companies that come for placement. Every graduate G has a preference list of companies where the graduate ranks the company that he/she is interested in. Note that the preference list for a candidate G may include only a subset of companies that come for placement. Every company C that participates in the placement has an upper bound on the number of graduates that it can hire. Let us call this upper bound the *capacity* of the company. Every company uses its own metric to evaluate the candidates who are interested in the company (this may be based on spot tests, CGPA etc.). Based on its metric for evaluation, each company has a preference list for graduates that are interested in this company where it ranks the interested graduates. Let the companies be denoted by $C_1, ..., C_m$ and the graduates by denoted by $G_1, ..., G_n$. Our software should reasonably *match* graduates to companies. In other words, it should assign graduates to companies such that certain constraints are satisfied. Let $f : \{G_1, ..., G_n\} \rightarrow \{C_1, ..., C_m\} \cup \{\perp\}$ denote an assignment function. That is, $f(G_{10}) = C_2$ iff graduate G_{10} has been assigned company C_2. A graduate G is not assigned any company iff $f(G) = \perp$.

Let us now see the constraint that such an assignment function should satisfy:

- At most one job: Each graduate can be assigned at most one company. This constraint is automatically taken care of by using an assignment function for matching.
- Capacity constraint: Each company cannot hire more that its capacity (which it had declared before placement started).
- Stable matching: The assignment f should be stable. An assignment is stable iff there does not exist any *blocking pairs*. A company-graduate pair (C, G) is said to be blocking iff **all** the following conditions are satisfied:
  1. G is interested in company C. That is, C appears in the preference list of graduate G.
  2. $f(G) = \perp$ OR G prefers C to $f(G)$.
  3. C's capacity is not been met OR C prefers G to at least one graduate assigned to it.

The *stable matching* constraint makes sense because if a blocking pair (C, G) exists, then G may leave its currently assigned company and move to C.

The problem we have described above is actually a popular problem known as Hospital Resident problem (HR problem in short). This is general version of the more popular Stable Matching problem. The HR problem may be solved using an extended version of the Gale-Shapley algorithm. You are requested to do research on these topics by yourself. There is a lot of material that is available on the web on these topics.

You are supposed build a software that helps T&P during the placement season. Your software should be able to store and access information regarding all companies and graduates participating in the placement and it should be able to suggest a stable assignment. As in the previous programming homework, your software will have to process a sequence of instructions. The efficiency of your software will crucially depend on the choice of data structures used. You are supposed to implement as efficient software as you can. This time, your submission, will also be evaluated for efficiency in addition to being evaluated for correctness. The faster you are able to process all requests, the higher the points you will get. You are completely free to decide the details of your implementation.

Given the problem statement, it makes sense to define classes for company and graduate. The outline of the company and graduate class is given below (feel free to add field and methods as per need).

```
class Company{
    String companyID;//The unique company ID given to every participating company by T&P
    String companyInformation;//Name and other basic information about the company
    int capacity;//This is the hiring capacity of the company
}
```

```
class Graduate{
    String graduateID;//This could be entry number or any other ID
    String graduateName;//Name of the graduating student
    double CGPA;//CGPA of the graduate
}
```

Task: 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/requests are discussed next.

- Add a company: This adds a new participating company into the system. Note that all the participating companies should be added before performing any other instructions. So, if there are 10 participating companies, then the first 10 instructions should be to add these 10 companies. You may assume that at most 1000 companies will participate in the placement this year. Examples of this instruction is given below:
  ```
  ADD COMPANY C1, Google, 5
  ADD COMPANY C2, Amazon, 10
  ADD COMPANY C3, Microsoft, 7
  ADD COMPANY C4, Flipkart, 15
  ```
  (`Google` is given companyID `C1` and its hiring capacity is 5).
- Add a graduate: This adds a graduate to the system. These instructions should succeed the previous instructions (add a company) and precede all other instructions. You may assume that at most 5000 graduates will participate in placements this year. Examples of this instruction is given below:
  ```
  ADD GRADUATE 2013CS10001, Aakash, 9.1, C1, C2, C3
  ADD GRADUATE 2013CS10002, Aastha, 9.3, C2, C1, C3
  ```
  (`Aakash` is a graduating student with entry no. `2013CS10001` and CGPA `9.1`. His first preference is `Google` followed by `Amazon` and `Microsoft`)
- Add company's ranking of interested graduates: Every participating company uses their own metric to evaluate interested graduates and ranks them (in decreasing order of preference). This ranking should be entered into the system. Example of this instruction is given below:
  ```
  RANK GRADUATES C1, 2013CS10001, 2013CS10002
  RANK GRADUATES C2, 2013CS10002, 2013CS10001
  ```
  (`Google` prefers `Aakash` to `Aastha` whereas `Amazon` prefers `Aastha` to `Aakash`)
- Display company information: This is self-explanatory. Here is an example:
  ```
  DISPLAY COMPANY C2
  ```
  This should produce the following output (on standard output):
  ```
  Amazon, 10
  ```
- Display graduate information: This is self-explanatory. Here is an example:
  ```
  DISPLAY GRADUATE 2013CS10001
  ```
  This should produce the following output (on standard output):
  ```
  Aakash, 9.1, C1, C2, C3
  ```

- Display graduate ranking by a company: It would nice to be able to query the ranking of the interested graduates done by a given company. Here is an example of this instruction:
`DISPLAY RANKING C1` This should output (in standard output):
`2013CS10001, 2013CS10002`
- Update the hiring capacity of a company: Sometimes during the hiring process while evaluating our graduates, the company may change its mind and decide to increase (or decrease) its hiring capacity. Here is an example of this instruction:
`UPDATE CAPACITY C1, 7`
(`Google` decides to increase its capacity to 7)
- Update CGPA of a graduate: Sometimes during the placement, the CGPA of a graduate may be updated due to a grade correction in some course. Here is an example of this instruction:
`UPDATE CGPA 2013CS10002, 9.4`
(`Aastha`'s CGPA gets updated to 9.4)
- Update preference of graduate: A graduating student may change his/her preference regarding companies. Our system should be able to allow this. Here is an example of this instruction:
`UPDATE GRADUATE PREFERENCE 2013CS10001, C2, C1`
(This should change the preference of `Aakash`. He now prefers `Amazon` to `Google` and he is not interested in `Microsoft`)
- Delete company: Sometimes a company may chose to drop out of the placement or IIT decides to disallow a company from participating in the placement due to some unavoidable circumstance. Then we should be able to delete a company from our system. Here is an example of this instruction:
`DELETE COMPANY C4`
(`Flipkart` should be deleted from the system)
- Delete graduate: A graduate may choose to drop out of placement due to various reasons. In that case, we should delete the graduate from our system. Here is an example of this instruction:
`DELETE GRADUATE 2013CS10002`
(`Aastha` opts out of placements and hence should be deleted)
- Perform assignment: Given the current preferences of graduates and companies, the extended Gale-Shapley algorithm should be executed to find a stable matching. The resulting assignment should be produced on the standard output. Here is an example:
`MATCH`
Suppose there are two companies with companyID's C1 and C2 and two participants P1 and P2. C1 prefers P1 to P2 and C2 prefers P2 to P1. Also, P1 prefers C1 to C2 and P2 prefers C2 to C1. Then a stable matching is P1-C1 and P2-C2. In this case the output should be:
```
P1, C1
P2, C2
```

Here is an example file `Query.txt`:

```
ADD COMPANY C1, Google, 5
ADD COMPANY C2, Amazon, 10
ADD COMPANY C3, Microsoft, 7
ADD COMPANY C4, Flipkart, 15
ADD GRADUATE 2013CS10001, Aakash, 9.1, C1, C2, C3
ADD GRADUATE 2013CS10002, Aastha, 9.3, C2, C1, C3
ADD GRADUATE 2013CS10003, Akshay, 8.9, C4, C1
RANK GRADUATES C1, 2013CS10001, 2013CS10002, 2013CS10003
RANK GRADUATES C2, 2013CS10002, 2013CS10001
RANK GRADUATES C3, 2013CS10001, 2013CS10002
RANK GRADUATES C4, 2013CS10003
MATCH
UPDATE GRADUATE PREFERENCE 2013CS10003, C4
DELETE COMPANY C4
MATCH
```

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

```
2013CS10001, C1
2013CS10002, C2
2013CS10003, C4
2013CS10001, C1
2013CS10002, C2
2013CS10003,
```

For any illegal query 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.