A report on

**FACULTY COURSE ASSIGNMENT AND OPTIMIZATION**

**By: Meghana S Seshasai (2021B5A72517G)**

**Anushka Sethi (2021B3A71074G)**

**Chandrika Singh Jadon (2021B1A72939G)**

**PROBLEM STATEMENT:**

The objective of this problem is to develop an assignment scheme that maximises the total number of courses assigned to faculty while aligning with their given preferences and the category-based constraints ("x1," "x2," "x3").

Faculty in each category are assigned different course loads, with "x1" handling 0.5 courses per semester, "x2" taking 1 course per semester, and "x3" managing 1.5 courses per semester.

When a course is shared between two professors, each professor's load is considered to be 0.5 courses.

**ASSUMPTIONS:**

* We assume that there is no preference order given for the course allotment to the faculty, so we arrange them randomly for allotment.
* Each course is present in the preference list of at least 1 faculty.
* Each course has to be assigned completely. For e.g. If only 0.5 of a course is assigned to a faculty and the remaining 0.5 cannot be assigned to any other faculty, then this is not a valid assignment.
* Each faculty can be assigned less than or equal to their course limit. For e.g. a faculty of category x2 can be assigned a total course load <=1.

Here we are maximising the no. of available courses to the students even if some of the faculty’s load is partial.

**LOGIC FOR THE CODE:**

The input is taken from a CSV file, which has a list of faculty details such as their name and preference order of courses.

Since we don’t have a given list of courses, we go through the preference list of all the faculties and we make a unique list of courses. We then make a map having keys as course names and values as popularity of the course (the number of times it occurs in the preference lists of the faculties).

We also make a list of faculties while going through the CSV file.

Then, we sort the collection of courses in the ascending order of popularity. The idea behind this is to assign the less popular courses first since they are harder to assign, then the more popular ones can be assigned easily. This maximises the total number of courses which can be taken in a semester.

Now, we create a local list of faculties and courses. The list of faculties is shuffled randomly so that we get a new sequence of faculties for every assignment.

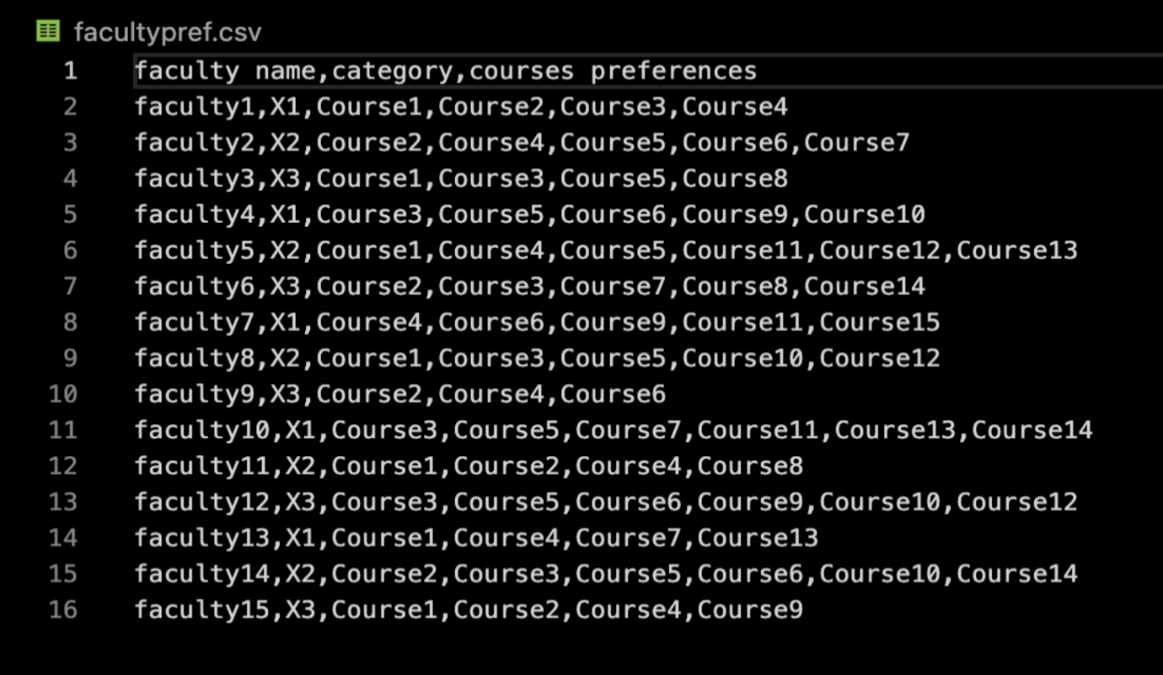
We start with the first course that is the least popular course and take the first faculty from the shuffled list of faculties. We see which category the faculty belongs to, check the number of empty slots that they have and assign the course accordingly.

Every assignment of the course to the faculty is 0.5. For an X2 faculty, who has 2 empty slots, the assignment is done twice and the course load becomes 1.

Then we repeat the sequence of steps for the rest of the courses in the local course list. Thus, we complete one iteration of valid assignment. 100 such iterations are carried out and stored in the Faculty-Course Assignment list. This list is sorted in descending order of total number of courses assigned in that iteration.

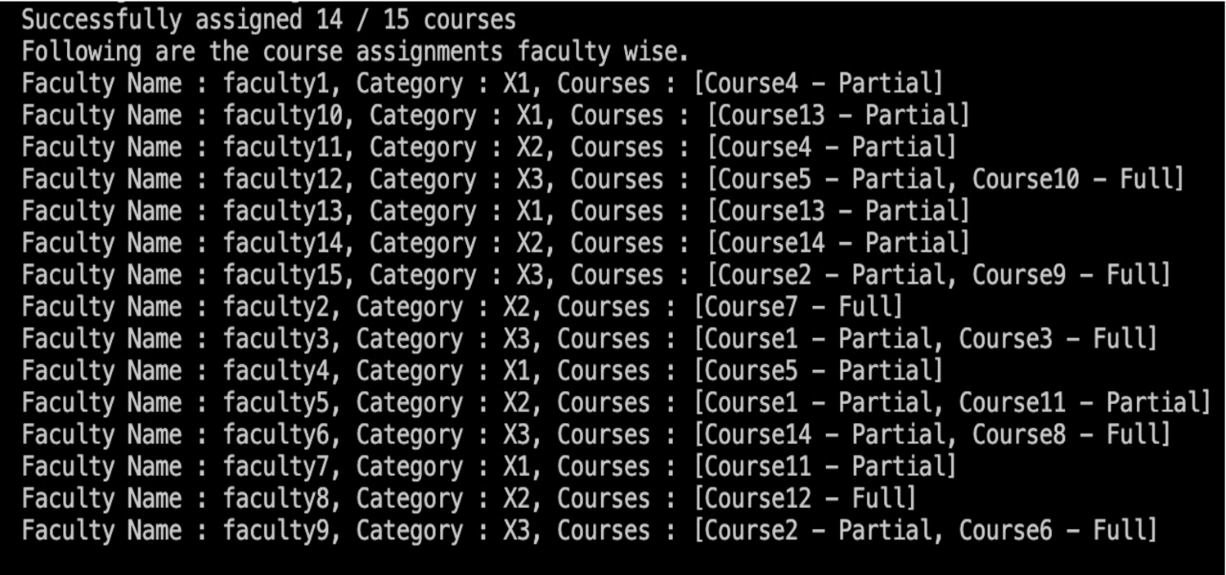
We take the first element of this list i.e. the one with the maximum number of courses allotted. This will be the best possible assignment and hence it is printed as the output.

**INPUT :**



This is the input csv file.

**OUTPUT:**

****

The is the output obtained on the terminal on running the CourseAssign file.

Here, a partial assignment is an assignment of course Load = 0.5

A full assignment is an assignment of course Load = 1

**FUTURE SCOPE:**

* Creating categories for courses such as core courses, electives and minor courses.

Core courses should not remain unassigned.

* Extending the number of professor categories beyond the existing three such as assistant and associate professors.
* Creating a separate list of courses for even and odd semesters and courses which are offered during the entire year.
* Implementing features that allow the system to adapt dynamically to changes, such as new faculty members, course cancellations, or unexpected faculty unavailability.
* Evaluate different existing optimization algorithms or heuristics to solve the assignment problem more efficiently.