DISCO-GRAPH OPTIMIZATION REPORT

The research problem involves optimizing a Course Assignment System with three faculty categories ("x1," "x2," "x3") having different course loads of (0.5,1,1,5) respectively.. Faculty members can share courses,
and each maintains a preference list. The goal is to produce maximum number of results while keeping in mind the category contraints and optimising the assignment of courses based on the preference list of each faculty.
A professor has to make a prefernce list for CDC's and Electives while giving 6 preferences in cdcs and 4 preferences in electives.

It is desired to extract input from a txt file and produce the output in the same. Keeping all the criteria in mind, C++ has been used in this project.

The provided C++ code is an implementation of an algorithm for assigning courses to professors based on their preferences and availability. The algorithm is divided into two main functions, assigncoursesCDC and assigncoursesEL,
each handling different types of courses: Core (CDC) and Elective (EL).
The parameters for the above functions have been extracted from txt files. Initially, proffessors of each category (x1,x2,x3) have been extracted from an input file into string vectors using fstream library and getline method.
The professors from all categories have then been inserted in another string vector named profs. Similarly, vectors for CDCCourses and ELCourses have been created.

The next step involves pairing CDCCourses as well as ELCourses with their preference number for each professor (Lowest mumber indicating highest preference to a course and 0 indicating that the professor is not interested in
teaching the course). The following has been obtained by extracting preference order for CDCCourses and ELCourses from different txt files and creating two vectors graphCDC and graphEL for storing the above mentioned relation.

Hence, the function assigncoursesCDC is called. The function assigncoursesCDC takes input vectors representing professor categories (x1, x2, x3), a list of professors (profs), core courses (coursesCDC),elective courses (coursesEL),
preference graphs (graphCDC and graphEL) and the total number of professors. It aims to

optimize the assignment of core courses to professors.The algorithm calculates the average preferences of each professor and identifies
the course with the highest average preference. It then selects professors with the lowest preference for the chosen course and creates pairs of professors for potential collaboration. The function recursively calls itself
with updated course and professor lists until all courses are assigned or no more professors are available.

assigncoursesCDC function is based on the fact that all CDCs must be assigned compulsorily. Therefore it is valid to first assign the CDCs and then the Electives. Furthermore, the CDC preffered least as a teaching option by the
professors should be assigned first to provide for maximum number of solutions. To obtain the same, a double vector avpref has been created. It stores the average preference of each course (higher the average preference number,
lesser the demand for teaching the course as lower preference number indicates higher priority). Therefore, the CDC preferred least has the highest avpref. For the obtained course, a for loop is run to find the professors that have
given the mentioned course least preference number (or highest priority). A vector of interested profs is created from the obtained list using graphCDC. Further all possible pairs are formed from interested profs that can teach
the course by dividing the load. The pairs are formed keeping in mind the category each professor belongs to and it has been ensured that no pair repeats. Furthermore, if a professor belongs to x2 or x3 category, he can teach
the course alone denoted by a pair having the same prof. After obtaining all the pairs, x1,x2,x3,profs,courses list are updated by storing them in variables
newx1,newx2,newx3,newprofs,newcourses. This is done by running a loop
over all the pairs, checking the categories of the first and the second element of the pair and making changes to the new variables accordingly. newcourses vairiable just erases the course that has been assigned.

It is to be noted that the code requires every professor has to provide atleast one preference. Otherwise the code will get an error that has been handled by the try block.

Next step checks whether all CDCs have been assigned or not. If not, the function recursively calls itself with the updated variables. If all CDCs have been assigned, the function calls assigncoursesEL for assignmnet of Electives.
The assigncoursesEL function focuses on assigning elective courses. It follows a similar logic to the core assignment but with slight variations in handling professor categories and collaboration pairs.
assigncoursesEL takes newx1,newx2,newx3,newprofs,ELcourses and graphEL as input parameters. Unlike CDCs, not all electives have to be assigned. Therefore the first elective to

be assigned should be the elective with lowest avpref
(or highest priority). avpref is calculated the same way as before and a vector of interested profs is genearted (the only difference being CDC was the least preffered course among CDCs and EL is the most preferred among electives).

The pairs are generated in a similar fashion and all the variables are updated. The function stops when there are no more available professors. THe function also recursively calls itself if there are still electives left to be
assigned.

The output is hence received in txt format by printing all possible pairs that had been stored as a vector pair containg CDC and another pair of professors. It is to be noted that a professor teaching a course a;one is represented by
a pair with both the elements same.

Hence all possible optimised solutions are generated.