***OMSCS 6310 – Software Architecture & Design***
***Project #1 [150 points]: Classroom Size Management System for a Local University***
***Summer Term 2016 – Prof. Mark Moss***

***Test Cases:*** These test cases are intended to help you ensure that you understand the basic principles of the project scenario, including the linear programming concepts of minimizing the "largest classroom size". The test cases are deliberately small, so that they can be reasonably worked through and verified by hand. Similarly, testing with smaller cases gives you an easier opportunity to test and troubleshoot your code: validating large test cases, and determining if you have errors when using those larger test cases are both significantly more difficult tasks.

We will use these test cases on your program, as you are also encouraged to do to ensure that your system is operating correctly. We will also use other test cases to check your program after submission, though, so please feel free to generate test cases of you own to cover other scenarios and/or "edge cases" that you feel are not covered here.

***Test Case #1:*** Consider the following student requests:
- 4 distinct students ($S_1$ through $S_4$) all taking Course 1

***[Feel free to stop reading here for a moment to see if you can determine the answer yourself]***

***Discussion:*** All four students are taking the Intro to Operating Systems Course, which is only offered in the Fall Semester. Given the four-year window from 2016 through 2019, there are four Fall Semesters available for the students who need to take the course. Ideally, this allows us to schedule the students so that they each take the course individually, as follows:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$, $C_1$ | | | $S_2$, $C_1$ | | | $S_3$, $C_1$ | | | $S_4$, $C_1$ | | |
| Fall | Spring | Summer | Fall | Spring | Summer | Fall | Spring | Summer | Fall | Spring | Summer |
| ***2016*** | | | ***2017*** | | | ***2018*** | | | ***2019*** | | |

Intuitively, spreading out the requests for a class over the available semesters as much as possible tends to minimize the "largest classroom size" value.
***Answer:*** X = 1

***Test Case #2:*** Consider the following student requests:
- 5 distinct students ($S_1$ through $S_5$) all taking Course 7

***[Feel free to stop reading here for a moment to see if you can determine the answer yourself]***

***Discussion:*** All five students are taking the Health Informatics Course, which is only offered in the Fall Semester. This problem seems very similar to Test Case #1, with the distinction having five students instead of four. This makes a big difference, however: the fifth student does not have a "free/empty semester", and must share with one of the other students. This mathematical concept is

sometimes referred to as the "Pigeonhole Principle": placing ***n + 1*** pigeons into a shelter with ***n*** pigeonholes will result in at least one pigeonhole with 2 or more pigeons.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1, C_7$ | | | $S_2, C_7$ | | | $S_3, C_7$<br>$S_5, C_7$ | | | $S_4, C_7$ | | |
| Fall | Spring | Summer | Fall | Spring | Summer | Fall | Spring | Summer | Fall | Spring | Summer |
| ***2016*** | | | ***2017*** | | | ***2018*** | | | ***2019*** | | |

Though I arbitrarily displayed Student 5 taking the course in Fall 2018, any reordering of the students and terms would have resulted in at least one of the Fall Semesters having 2 or more students.
***Answer:*** X = 2

***Test Case #3:*** Consider the following student requests:
- 4 distinct students ($S_1$ through $S_4$) all taking Course 1
- 1 distinct student ($S_5$) taking Course 10

***[Feel free to stop reading here for a moment to see if you can determine the answer yourself]***

***Discussion:*** Once again, this test case is very similar to Test Case #1. The distinction here seems to be that Student 5 is taking the Computer Vision Course, which is offered in the Spring Semester. Initial thoughts might lead you to believe that this means there is no effect on the Course 1 students – until you remember that Course 10 has Course 1 as a prerequisite. This puts us back in the same situation as Test Case #2:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1, C_1$ | | | $S_2, C_1$<br>$S_5, C_1$ | | | $S_3, C_1$ | $S_5, C_{10}$ | | $S_4, C_1$ | | |
| Fall | Spring | Summer | Fall | Spring | Summer | Fall | Spring | Summer | Fall | Spring | Summer |
| ***2016*** | | | ***2017*** | | | ***2018*** | | | ***2019*** | | |

Once again, I arbitrarily selected a semester (Spring 2018) for the Computer Vision Course where the main constraint is that it must follow the Intro to Operating Systems Course.
***Answer:*** X = 2

***Test Case #4:*** Consider the following student requests:
- 4 distinct students ($S_1$ through $S_4$) all taking Course 15

***[Feel free to stop reading here for a moment to see if you can determine the answer yourself]***

***Discussion:*** This test case appears similar to Test Case #1 on the surface, until we consider the distinction that the High Performance Computing Course (15) has the High Performance Computer Architecture Course (5) as a prerequisite. Course 15 is a Fall Only course, while Course 5 is a Spring Only course, and this constraint has a significant effect on our scheduling efforts. Since prerequisites must be taken before their dependent courses, then there is no feasible way that any of the students

can take Course 15 during the Fall 2016 Semester – there is no Spring 2015 Semester (or earlier) available for them to take Course 5:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $S_1$, $C_5$ | | $S_1$, $C_{15}$ | $S_2$, $C_5$<br>$S_3$, $C_5$ | | $S_2$, $C_{15}$ | $S_4$, $C_5$ | | $S_3$, $C_{15}$<br>$S_4$, $C_{15}$ | | |
| Fall | Spring | Summer | Fall | Spring | Summer | Fall | Spring | Summer | Fall | Spring | Summer |
| ***2016*** | | | ***2017*** | | | ***2018*** | | | ***2019*** | | |

This is just one possible schedule, but the alternative schedules don't provide any improvements on the answer. The Pigeonhole Principle helps our reasoning here as well: because the Fall 2016 Semester can't be used for Course 15, then those four students have only three remaining Fall Semesters to take the course; and, consequently, at least one of those Fall Semesters must have 2 or more students.
***Answer:*** X = 2

***Test Case Summary:*** Please remember that you can complete Project #1 successfully without becoming a linear programming expert; however, understanding some of the basic principles can be very helpful while designing, implementing and testing your system. The cases above are deliberately simple to facilitate explanation, but could become much more complicated very quickly as more students are added, and as the course selection varies more widely. As a completely optional challenge, determine if a feasible schedule exists for a single student taking all 18 courses.

Please don't be "lured" into trying to implement your own solutions for the general scheduling and minimization problems – let Gurobi work for you. Focus on identifying all prerequisites, and then generating the correct constraints in Gurobi format.

***Test Case Files & File Formats:*** The test case files are all comma separated value (CSV) text files, with multiple lines in the format:
**`Student_Demand_<TestCaseIndex>: <StudentID, CourseID>`**

- StudentID values will be integers from 1 through 400 (inclusive), and will vary per test case.
- Course ID values will be integers from 1 through 18 (inclusive) per the Course Catalog.
- Each line <X, Y> will represent a request from Student X to take Course Y.
- Student requests will be listed contiguously: all of Student 1's course requests, followed by all of Student 2's course requests, etc. until the End of File marker.

Other static files that you are welcome to use (will be the same for all test cases during Project #1):
**`Course_Titles: <CourseID, CourseName>`** as listed in the Course Catalog.
**`Course_Availability: <CourseID, SemesterCode>`** where SemesterCode has the values 0, 1 or 2 for the Fall, Spring or Summer Semesters, respectively. Multiple lines will be used for courses that are offered in multiple semesters.
**`Course_Prerequisites: <BeforeCourseID, AfterCourseID>`** where BeforeCourseID is a prerequisite course for AfterCourseID.