Joseph Lee
Module 4 Basic Programming Writeup

I created five private methods for this problem. I have three of them to generate v rows of the matrix, where v is the number of courses plus number of students plus one source plus one sink. The methods are called createAdjacencyMatrix and its helper methods createStudentRow and createCourseRow. The createAdjacencyMatrix takes in the list of students, list of courses, the map of students and their courses, and the map of courses and capacity of courses. createAdjacencyMatrix then uses the lists and maps to find the relationships between each of them and set the values. The relationships can be: node and itself (that has value of 0), student and a course (1 if the student register the course and 0 if not), source and the students (where value is same as classes enrolled), a course and a sink (where value is equal to given in the input and stored in the map), and anything else is set to 0. Two other private methods (createStudentRow and createCourseRow) were created as a helper method for createAdjacencyMatrix. The method called callFordF is used to find the maximum flow. It first clones the adjacency matrix to create a matrix for storing residuals and it loops through while there is a path from source to sink, which is checked by calling the BFS method. I have two loops in callFordF methods: the first for loop is traversing through the path to find the min residual capacity then set it to the current flow and the second for loop that sends backflow in the opposite direction then increment the residual for rotated coordinates. For the BFS method, I created a priority queue to check if there is a path from source to sink. The loop checks if the given coordinate has not been passed and the residual is greater than 0 and if it reaches the sink then it sets the previous value to the row coordinate and returns true. Otherwise the method returns false. After maxFlow is found, I check if the maximum flow is the same as the number of students times the number of classes the student should enroll. If the maxflow matches the number then I save “Yes” into the solution and otherwise store “No.” At the end, the time complexity will be O(SCV^2) where S is number of students and C is number of courses a student enrolls, and as the Ford Fulkerson is O(f*E) and E is equal to V^2, which is from BFS.