Jeff Beaupre

ICS-340

Program 3

Justification and Explanation of Design

My program used a divide and conquer greedy algorithm with sorting. I read in all classrooms and sort them by room type into separate array lists, then sort by capacity, lowest to highest. I then read in the Course file and sort them by room type into separate array lists, then sort those by enrollment numbers, lowest to highest. To create the schedule, I begin with assigning the computer and science courses, these need to be in a specific room thus their added to the schedule first. For these two course types the process is as follows:

* For each course in the computer/science course List
  + Look to see if the courses preferred location matches the classroom location
  + Look to see if the enrollment fits into the classroom capacity
  + Assign to a room if above conditions are met
* Check to see if any courses are left over if so:
  + Sort the unassigned list by lower division classes first (this ensures that a lower division compatible room is not taken by an upper division class first)
  + Check to see if the course is lower division
    - If lower division, look at only lower division compatible campus
    - If not lower division, look at any available classroom at different campus
  + Assign the course to an available room
  + Re-check to see if any other courses are still unassigned

The process for the Regular type course is close to being the same as above, but with the addition of an unassigned course will look at any available computer and science room on the same campus before looking to a different campus, this will preserve the soft constraint on regular courses, this constraint would be ignored on science and computer courses if they cannot find a compatible room on their requested location.

I chose this style of algorithm because initially I wanted to do bipartite matching, but found that the implementation would be a bit difficult for this application. I felt strongly enough that by sorting classrooms/course into separate arrays by type, while also sorting by size, I would be able to avoid any mismatching. A class would be assigned to the first room it was able to be set to, and would not take a spot of a different class unless absolutely necessary. Furthermore, the act of dividing up the classrooms/courses by type, made for an easy solution for the given constraints, the science and computer courses could not be in any other room, so dealing with them first was easy and left the regular courses free to look at the rest of the classrooms at the campus if needed.