Lew, Phoebe

CSCI 103

Crowley, TTH

Design write-up PA3

**System Name:** Scheduling Classes

**System Description:**

This system reads a text file containing building and course information. It uses this data to assign each course to a room, while avoiding conflicting classes. After assigining all courses (if possible), it outputs the schedule according to both course number and classroom.

**Global Data:**

Structs (5), they are dynamically allocated in main():

Room - stores room specifications

Course - stores course specifications

Section, Section\_Constraints - stores section specifications

Scheduled Sections - stores courses with their corresponding room data

Other Variables (8):

npos - used to determine of MWTR are found in input text file

numRooms - counts number of rooms

lineIndex - counts number of sections

scheduledCount - tracks how many sections have been thus far scheduled

\*rooms, \*courses, \*allConstraints, \*scheduledSections - pointers to arrays to be used in scheduling functions (provided by crowley)

**Function/Class Descriptions:**

Function: main

Parameters and types: none

Return type: int

Function description:

Function: sortWay

* Parameters and types: takes in const Room &lhs, const Room &rhs
* Return type: bool
* Function description: This function is used in sortRooms(). It will sort in descending rather than ascending order. It determines whether the first room or second room has a greater capacity.

Function: sortRooms

* Parameters and types: none
* Return type: none (void)
* Function description: sortRooms() uses sort(), which takes in three parameters. The first is the starting address to begin sorting, the second is the ending address, and the third defines how sort() will actually perform the search, in this case in descending order.

Function: roomsMatch

* Parameters and types: Room scheduledRoom, int scheduleRoomIndex
* Return type: bool
* Function description: roomsMatch() first declares a bool variable as false. It checks to see if both the building code and room number of a course to be scheduled matches any of the previously scheduled sections. If they match, i.e there is a possible conflict, the function returns true.

Function: daysMatch

* Parameters and types: int\* scheduledDays, int\* constraintDays
* Return type: bool
* Function description: daysMatch() does essentially the same thing as roomsMatch(). If we find that the room and building for two sections match, this function checks to see if we're using the room on the same day. It uses a for loop to check all four days of the week. If any one of the days match, the bool variable is set as true and we stop searching through all the days.

Function: fixTime

* Parameters and types: int constraintStartTime, int numMinutes
* Return type: int
* Function description: This function takes the start time of a section and calculates the end time, which then becomes the time when a new section can be scheduled to avoid conflict. The while loop counts how many one hour sections are in the number of minutes in the class so the new start time will be correctly changed. This avoids the new start time to be something like 3:80pm.

Function: timesOverlap

* Parameters and types: int scheduledStartTime, int scheduledEndTime, int constraintStartTime, int constraintEndTime, int numMinutes
* Return type: int
* Function description: Given that scheduled is false, this function checks if the scheduled start time is earlier than the constraint start time and the scheduled end time is later than the constraint start time, or if the scheduled start is earlier than the altered start and scheduled end time is later than the altered end times (altered via fixTime()). If so, we cannot schedule the new section at the given time because there is an overlap. We move the start time later by 10 minutes and recheck. If there is no overlap, this is a possible start time and this time is returned by the function.

Function: scheduleAvailable

* Parameters and types: Section sectionToSchedule, int roomIndex
* Return type: bool
* Function description: scheduleAvailable takes in a section to schedule and tracks which room is being used. It runs while we haven't scheduled all the sections. It checks to see if the section can be scheduled in the rooms. Then, it makes sure that there won't be a conflict with previously scheduled sections. If we are unable to schedule the section in the first room, it moves onto the second room and re-runs the checks against room and days. If we are able to schedule the section without conflict, we write all the course, room, and constraints data to the scheduledSections[] array. This array will eventually be used to print out all sections that were scheduled.

Function: scheduleSections

* Parameters and types: none
* Return type: bool
* Function description: scheduleSections() is the main function to call all the functions. After having ordered the rooms from largest to smallest, this function takes a section and room and tries to schedule them by calling scheduleAvailable(). If we cannot schedule the after checking all the rooms, the function returns false. Otherwise, it returns true.

Function: classWay

* Parameters and types: const Scheduled\_Sections &lhs, const Scheduled\_Sections &rhs
* Return type: bool
* Function description: Used to sort the scheduledSections array by class prefix

Function: classWay2

* Parameters and types: const Scheduled\_Sections &lhs, const Scheduled\_Sections &rhs
* Return type: bool
* Function description: Used to sort the scheduledSections array by class number.

Function: sortClass

* Parameters and types: const Scheduled\_Sections &lhs, const Scheduled\_Sections &rhs
* Return type: none (void)
* Function description: Sorts the scheduledSections array first by prefix and then by course number.

Function: roomWay

* Parameters and types: const Scheduled\_Sections &lhs, const Scheduled\_Sections &rhs
* Return type: bool
* Function description: Used to sort the scheduled sections array by the building code.

Function: roomWay2

* Parameters and types: const Scheduled\_Sections &lhs, const Scheduled\_Sections &rhs
* Return type: bool
* Function Description: Used to sort the scheduledSections array by room number.

Function: sortRoom

* Parameters and types: const Scheduled\_Sections &lhs, const Scheduled\_Sections &rhs
* Return type: none (void)
* Function Description: Sorts scheduledSections first by building code and then by room number.

**Compiling and Testing**

Tell us how to compile your program and run it.