CSC 415 Spring 2024

Assignment 1 – Application Development

Due: Thursday February 8, 2024, by 11:59 p.m.
Grade: 100 points as per the rubric; 10 points penalty for each day late.

You must complete, i.e., design and implement the solution for, this assignment **individually**.

Objectives:

The main goals of this assignment are for you to reinforce your own understanding of fundamental computer science programming concepts and problem-solving, and to learn a new programming language, Ruby. Through your submission for this assignment, you must demonstrate:

- An ability to design and implement a computational solution for a problem, based on specified requirements and constraints.
- A strong understanding of fundamental programming concepts such as abstraction, classes, iteration, and modularity.
- An ability to determine and implement data structures and algorithms appropriate for a solution.
- An ability to independently learn a new language, specifically Ruby, and implement an efficient, and reliable program in that language.
- An ability to learn and use git, GitHub, and the individual virtual machine (VM) assigned to you.

Program Requirements:

Assume that you are a software engineer at GF Software Solutions, Inc., and that your company develops software solutions for public higher education institutions, like TCNJ, on a pro bono basis.

Problem:
Every year, student organizations plan large events that may require multiple spaces, of different sizes and capacities, in close proximity to each other. An example is HackTCNJ, TCNJ's hackathon, which requires a large room for the opening and closing sessions, a room where teams can showcase their projects, and rooms for workshops and team collaboration. Some of these rooms may be needed for short periods of time, while others may be needed for the entire duration of the event. Finding and scheduling a group of spaces that meet various constraints can be challenging, and a well-designed algorithm would alleviate the problem.

Specifications:
GF Software Solutions, Inc. has been awarded a contract to develop an application that will help student organizations in many of their activities. You have been assigned to develop a module, called **`Scheduler`**, that will schedule rooms for large events, based on availability, capacity, seating, and other characteristics. The algorithm is eventually expected to be general enough to be used at other institutions and for different types of events. To reduce the scope, and to help you understand the problem better, the focus here is on HackTCNJ, and the constraints described in this document.

For this assignment, you will design and implement a prototype of `Scheduler`, in Ruby, based on the description and functionality provided below. Be sure to develop the prototype such that in the future it can be expanded (scalable and reusable) to handle different types of rooms and events, and more complex constraints.

The `Scheduler` prototype must include user interaction and the functional requirements described below:

- Input data from files.
- Generate scheduling plan.
- Output scheduling plan(s) to files.
- Display scheduling plan(s) on the screen.

User Interaction: The `Scheduler` prototype must enable the user to enter details about the event to be scheduled, and the names of the input and output files.

Input data: Read data from the input files and store them in appropriate data structures. Choose data structures based on the type of data to be stored and the algorithms you will use. Be sure to consider time and space complexity as you make these decisions. Do not use a database.

The first line of the input file will identify the columns (or the fields) for the remaining rows. Do not read the first line into the data structure.

Each of the remaining rows will have values for the fields separated by comma; one row represents data for one item (e.g., rooms).

To reduce the complexity of this program, you can assume that input files will be formatted correctly, and that all fields will have data, i.e., no field will be empty. The actual data can be assumed to be in the correct format (data type) but the values may or may not be valid.

Format for the input file that contains the list of rooms on campus.

First row: Building, Room, Capacity, Computers Available, Seating Available, Seating Type, Food Allowed, Priority, Room Type

- Building: String; the building in which the room is located
- Room: String; the room number or name
- Capacity: Integer; the maximum number of occupants allowed
- Computers Available: String; whether or not the room has computers for participants; allowed values are 'Yes' and 'No';
- Seating Available: String; the type of furniture provided in the room
- Seating Type: String; whether the room is tiered or all seating is on one level; allowed values are 'Tiered' and 'Level'
- Food Allowed: String; whether or not food and beverages are allowed in the room; allowed values are 'Yes' and 'No'
- Room Type: String; the typical purpose of the room, e.g., Classroom, Conference Room, computer lab, etc.
- Priority: String; which department or program has priority for scheduling this room

Examples of rows in this file:

```
Forcina,403,30,Yes,Large Tables with Chairs,Level,No,Computer Science,Computer Lab
BSC,100,800,No,Chairs,Level,Yes,Administration,Event Center
```

Format for the input file that lists the rooms that are already reserved (i.e., they are not available).

First row: Building, Room, Date, Time, Duration, Booking Type

(Note: Only reserved rooms will appear in this file, i.e., if a valid room is not in this file for a desired date/time, then assume it is available.)

- Building: String; the building in which the room is located
- Room: String; the room number or name
- Date: Date; the date of room reservation; format is yyyy-mm-dd
- Time: Time; start time of the room reservation; format is hh:mm AM/PM
- During: Time; the length of time for which the room is reserved; format is hh:mm
- Booking Type: String; the type of event for which the room has been reserved, e.g., class, conference, event, etc.

Examples of rows in this file:

```
Forcina,403,2024-04-09,11:00 AM,1:20,class
BSC,100,2024-04-12,9:00 AM,5:00,event
```

Generate Scheduling Plan(s): The program must ask the user to enter the constraints for the event, and generate a scheduling plan with an appropriate set of rooms that are available to be reserved for the event, and meet the constraints. Higher priority should be given to finding rooms in the same building. There may be more than one possible plan.

Constraints to be entered are:

- Date of event; format is yyyy-mm-dd
- Start time of the event; format is hh:mm AM/PM
- Duration of the event; format is hh:mm
- Number of attendees

Additionally assume the following constraints:

- There will be a one-hour opening session, and all attendees must fit into one room.
- A one-hour meal must be provided for every six hours of the event.
- The meal must be served in more than one room (to reduce congestion).
- On average, 60% of the attendees will eat each meal.
- About 10% of the attendees will not have their own computers so these must be provided.
- There should be some rooms where attendees can work on their projects individually or in small groups; there may be more than one group in a room depending on the capacity.
- There will be a three-hour closing session for final demonstrations, judging, and awards, and all attendees must fit into one room.
- The larger room(s) used for the opening and closing sessions cannot be used for individual or small group work.
- The rooms should ideally be in close proximity to each other, i.e., in the same building.

Display Scheduling Plan(s): The program should display the plan(s) generated, neatly formatted, with the following information: Date, Time, Building, Room Name, Capacity, Whether computers are available, Whether food is allowed, Purpose for event (opening session, group work, etc.)

Write Scheduling Plan(s) to a File: The program must write the scheduling plan(s) with the list of suggested rooms in the plan(s), in a csv file; the filename should be specified by the user.

Format for the output file (.csv) that contains the scheduling suggestions:

First row: Date, Time, Building, Room, Capacity, Computers Available, Seating Available, Seating Type, Food Allowed, Room Type, Priority, Purpose

- Date: Date; format is yyyy-mm-dd
- Time: Time; start time; format is hh:mm AM/PM
- Building: String; the building in which the room is located
- Room: String; the room number or name
- Capacity: Integer; the maximum number of occupants allowed
- Computers Available: String; whether or not the room has computers for participants; allowed values are 'Yes' and 'No';
- Seating Available: String; the type of furniture provided in the room
- Seating Type: String; whether the room is tiered or all seating is on one level; allowed values are 'Tiered' and 'Level'
- Food Allowed: String; whether or not food and beverages are allowed in the room; allowed values are 'Yes' and 'No'
- Room Type: String; the typical purpose of the room, e.g., Classroom, Conference Room, computer lab, etc.
- Priority: String; which department or program has priority for scheduling this room
- Purpose for event (what the room can be used for, e.g., opening session, group work, etc.)

CSC 415 Spring 2024

General Assignment Guidelines

- For the assignment to be considered complete, and receive a passing grade, it must, at the very least, enable user interaction, read and write from user specified files, and implement a scheduling algorithm that provides some output.
- You may make assumptions about situations that are not described in these requirements. Document your assumptions in the source code where appropriate, as well as in a document called `SchedulerNotes.pdf`. See the Deliverables section below for more details on the content that should be included in this document.
- The program must be developed and demonstrated to work on the individual VM assigned to you, in the `student1` account.
- The program must be implemented in **Ruby** with an appropriate data structure that you will determine. Do not use Rails or a database for this assignment.
- Ensure that your program is modular, secure, reliable, and efficient.
- Use a good coding style with consistent indentation to ensure that your source code is readable.
- Source code files must include maintenance information, and the code should be documented to explain functionality and design decisions. See the general guidelines for programs and documentation, posted on the Canvas page "Guidelines for Programs".
- You must use git for version control, and GitHub to host the remote repository. Add Professor Pulimood (GitHub id: `tcnjcs`) as a collaborator for the repository on GitHub.
- Commit your source code frequently to GitHub to avoid inadvertent losses.
- The repository must be private. Making the repository public during the semester is a violation of the Academic Integrity policy.
- Do not commit input or output files to GitHub. Update `.gitignore` to omit these files when you commit changes to GitHub.
- Sample input files will be posted on Canvas closer to the due date. However, you must create a few (perhaps smaller) files of your own for development and testing purposes.
- Use the built-in Ruby CSV library to read data from csv files, and write data out to files. See https://ruby-doc.org/core-2.5.1/File.html for details on the `File` class methods and usage; see https://www.rubyguides.com/2018/10/parse-csv-ruby/ for details on parsing csv files.

Learning and Developing:

You are encouraged to ask general questions about the requirements, programming concepts, and errors in class or on Slack, and discuss questions specific to your solution during office hours, or through direct messages to Professor Pulimood on Slack. You are strongly encouraged to contact Ayesha, for technical help. Do not go to the tutors in the CS Department or the Tutoring Center.

See the page "Toolkit for SE - Ruby and Rails" on Canvas for useful resources to learn Ruby.

You may work with each other to learn the Ruby language, but you are not allowed to help each other, or use generative AI, to solve the assigned problem and implement the program. Seeking assistance from any source other than Dr. Pulimood or Ayesha, or providing inappropriate assistance to other students, are violations of TCNJ's Academic Integrity policy.

Program Demonstration and Grading:

After the due date, you must meet with Professor Pulimood for a technical review of your program, i.e., to demonstrate your program on your VM, and discuss your design and code. During this time, you will be given extensive verbal feedback about your program, that you will be expected to utilize for future assignments in this class. You should come prepared to take notes on feedback provided.

Make an appointment using the link provided on the assignment page on Canvas. Each appointment slot is for 20 minutes with a few additional minutes of buffer time.

Deliverables:

On Canvas in the "**Assignment 1**" dropbox, submit a file called `SchedulerNotes.pdf` with:

- Your name.
- Your VM name.
- Password for the `student1` account.
- The full file pathname for the directory where your program resides on your VM.
- The url for the GitHub repository for this program.
- Assumptions you have made.
- Instructions for running the program.
- Known bugs, issues or limitations.

Do NOT submit your Ruby code on Canvas.

On your VM, in the subdirectory named "`scheduler`" that you created in Exercise 4:

- All the Ruby source code files.
- All the input and output files you used and created during development and testing.
- The sample input files posted on Canvas.

On GitHub, in the repository named "`scheduler`" that you created in Exercise 4:

- All the Ruby source code files.
- Instructions for running the program.
- Known bugs, issues or limitations.

Do NOT submit `SchedulerNotes.pdf`, data files, VM name, or passwords on GitHub.