CSC 415 Spring 2021

Assignment 1 – Application Development

Due: Monday February 15, 2020 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 develop the solution for, this assignment **individually**.

You are encouraged to ask general questions about the requirements, design and programming concepts, or error messages in class or on Slack. You are encouraged to discuss questions or concerns specific to your solution during office hours, or through direct messages to the professors on Slack. You can also ask the course tutor, Heavenly, for technical help. You may work with each other to learn the Ruby language. Seeking assistance from any source other than the professors and course tutor, or providing assistance to other students, on solving the problem and writing the program are violations of TCNJ's Academic Integrity policy.

Objectives:

The main goals of this assignment are for you to demonstrate that you have a good understanding of prerequisite computer science concepts, and the ability to design and implement a computational solution for a problem based on the requirements provided. Specifically, you should demonstrate:

- An ability to solve a problem, based on specifications and requirements provided,
- A good grasp of fundamental programming concepts like iteration (loops), branches (selection), recursion, pointers, etc. as well as abstraction and classes,
- The ability to determine the appropriate data structure for a purpose, and to implement it
- At least a basic understanding of time and space complexity,
- The willingness and ability to learn and gain experience in a new language, specifically Ruby, by implementing a program to meet requirements,
- At least a basic facility in using git, GitHub, and the individual VM assigned to you.

Requirements:

Assume that you are a software engineer at GF Software Solutions, Inc., and that your company has been awarded a contract to develop an application that will help students schedule rooms for large events, such as the TCNJ hackathon, 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, but to help you understand the problem better, the focus here is on the TCNJ hackathon and the specific constraints present here.

HackTCNJ, the annual TCNJ hackathon, hosts student participants from multiple universities and institutions, and mentors from sponsoring companies and the College. The 24-hour event requires rooms to be available and scheduled without interruption for the entirety of the hackathon. The hackathon will need enough rooms to (1) handle the capacity of the students (2) allow for food and beverages (3) provide a two-hour long presentation of hacks with all attendees able to fit.

Using Ruby, implement an application that will use current data about rooms on campus, their characteristics and availability, and return a list of rooms and their respective times they can be used during HackTCNJ.

Implement a prototype of an application, in Ruby, with the functionality described below.

- ***Input data***: The program should allow the user to provide the names of two files that contain data in a specified format, read data from both files, and store the data in appropriate data structures for further processing.

  In each input file, the first line identifies the columns or the fields for the remaining rows. Each of the remaining rows will have the following fields separated by comma; one row represents data for one item. Assume that the input files will be formatted correctly.

File 1: Data about the rooms on campus; there must be a value in every field.

Building, Room, Capacity, Computers Available, Seating Available, Seating Type, Food Allowed, Priority, Room Type

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

File 2: Data about availability of rooms on campus; there must be a value in every field.

Building, Room, Capacity, Computers Available, Seating Available, Seating Type, Food Allowed, Priority, Room Type

- Building – the building in which the room is located
- Room – the room number or name
- Date – the date of room reservation; format is yyyy-mm-dd
- Time – start time of the room reservation; format is hh:mm AM/PM
- Duration – length of time for which the room has been reserved; format is hh:mm
- Available – whether the room is available or not; possible values are "true" and "false"; if the value is "false" then Booking Type will have a value
- Booking Type – the type of event for which the room has been reserved, e.g., class, conference, event, etc.

- ***Suggest Room Reservation Schedule*: The program must ask the user to enter the following:**
  - 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

  The program must assume the following constraints:
  - There will be an opening / welcome session (one hour), and all attendees must fit into one room.
  - A one-hour meal must be provided for every six hours of the hackathon.
  - The meal must be served in separate rooms.
  - 60% of the attendees will eat each meal.
  - 10% of the attendees will not have their own computers so we must provide them.
  - There should be some rooms for attendees to work on their projects individually or in groups; there may be more than one group in a room depending on the room size.
  - There must be more than one room used during the event.
  - There will be final presentations and awards, and all attendees must fit into one room.

The program must run your algorithm to find an appropriate set of rooms that meet the specified constraints, and output the list of rooms with:
- Date
- Time
- Building
- Room Name
- Capacity
- Computers Available
- Food Allowed

- ***Reserve Rooms*:** The program must enable the user to reserve some or all of the rooms listed in the output of the "Suggest Room Reservation Schedule" functionality, and then update room availability accordingly.

- ***List rooms*:** The program should display, neatly formatted, the final list of rooms that have been reserved for the event.

  The user must be able to request at any time a listing of all rooms in the system, along with their availability, neatly formatted.

- ***Write data to a file*:** The program must allow the user to specify the name of a file into which the updated information about the rooms and availability can be written.

The program must be hosted and demonstrated to work on the individual VM assigned to you, and should enable user-interaction, i.e. allow the user to select the tasks to be completed, specify names of files, etc. Sample input files will be posted on Canvas closer to the due date, but you must create a few of your own for development and testing purposes. You will be expected to demonstrate that the program works successfully with other files as well. You can assume that all input files will have the correct format, as specified above.

Use git for version control, and GitHub to host the remote repository. Add both, Professor Pulimood and Professor Devlin, as collaborators to the repository on GitHub. Be sure to commit changes frequently so that you do not inadvertently lose any of your code.

The intent of the assignment is for you to demonstrate your ability to learn a new language (in this case, Ruby), and implement a solution to a problem using this language. Therefore,
- You cannot use any built-in algorithms, or algorithms / code you find from other sources.
- You must implement any searching or sorting algorithms you might need.
- You must give consideration to time and space complexity, use appropriate data structures, and implement efficient algorithms.
- You are encouraged to use functions from the built-in Ruby CSV library to read data from files, and write data out to files.

Follow good programming practices to ensure that your program is modular, secure, reliable and efficient. Use a good coding style with consistent indentation to ensure that your source code can be easily read. **Source code files must be documented with maintenance information.** You must also document your code well to explain functionality and design decisions. See the general guidelines for programs and documentation, posted on the Canvas page "Guidelines for Programs".

Useful resources to learn Ruby:
- **`LinkedIn Learning`** – Free with your TCNJ id and password; access it from the TCNJ Today page at https://today.tcnj.edu/.
- **`codecademy.com`** – a good place to start is https://www.codecademy.com/learn/ruby.
- **`http://ruby-doc.org/`**– Documentation for the Ruby programming language.

After the due date has passed, you must meet with one of the professors individually to demonstrate your application on the VM, and to discuss your program, to verify that it is complete and works correctly to meet the requirements. **Make an appointment using the link that will be provided on the Canvas page for this assignment.** Each appointment slot is for 30 minutes though we expect that the actual time needed will be about 20 minutes. You should come prepared to take notes on feedback provided.

Deliverables:

On Canvas in the "**Assignment 1**" dropbox, submit a file called "readme.pdf" that specifies:

- Your VM name and password.
- The full file pathname for the program on your VM.
- The url for the GitHub repository.
- Instructions for running the program.
- Known bugs, issues or limitations.

On your VM, in an appropriate subdirectory:

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

On GitHub, in an appropriate repository:

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

Label each source code file and the readme file with your name and assignment number.