CS222: Data Structures and Algorithms
Summer Semester (May - Aug 2024)
Final Project

STATEMENT ABOUT ACADEMIC HONESTY AND INTEGRITY

Academic honesty and integrity are very important at Ashesi and central to the achievement of our mission: To train a new generation of ethical and entrepreneurial leaders in Africa to cultivate within our students the critical thinking skills, concern for others, and the courage it will take to transform a continent. As this mission is our moral campus, we recommend you take it seriously in this course without any exceptions at all.

Ashesi therefore does not condone any form of academic dishonesty, including plagiarism and cheating on tests and assessments, amongst other such practices. Ashesi requires students to always do their own assignments and to produce their own academic work unless given a group assignment.

As stated in Ashesi's student handbook, Section 7.4:

"Academic dishonesty includes plagiarism, unauthorized exchange of information or use of material during an examination, unauthorized transfer of information or completed work among students, use of the same paper in more than one course, unauthorized collaboration on assignments, and other unethical behaviour. Disciplinary action will be taken against perpetrators of academic dishonesty."

All forms of academic dishonesty are viewed as misconduct under Ashesi Student Rules and Regulations. Students who make themselves guilty of academic dishonesty will be brought before the Ashesi Judicial Committee and such lack of academic integrity will have serious consequences for your academic records.

OBJECTIVE:

This is the final project of your course, which involves applying and deepening your understanding of data structures and algorithms by developing a Java application addressing a real-world problem preferably related to **climate change**. You will use various data structures and algorithms to efficiently process and analyze data, implement core functionalities, and provide meaningful outputs via a Command Line Interface (CLI), though a GUI-based solution is encouraged.

PROJECT TASK:

You can choose one of the given application scenarios either related to **climate change** or other (see the end of the document for the same) and implement it using Java. Your application should use a combination of data structures and algorithms, and it must be fully functional.

***Note**: You can make use of the data structures and algorithms as implemented by you. Equally, you can also choose to use the data structures from the Java collections framework and any utility functions from the Java library.*

INSTRUCTIONS:

1) This is a group project. You will work in groups composed of **four** members. The project groups can spawn both sections A and B, though groups formed from the same section are preferable for ease of grading and managing the related works.
2) Groups should be formed by **16th July 2024, 5 PM** and the group composition must be submitted via Canvas. (if members are in a different cohort, indicate the same clearly).
3) Each group should assign roles to its members to ensure all aspects of the project are covered. Suggested roles include *Group lead* (oversees the project timeline, ensures deadlines are met, and coordinates meetings), *Developers* (responsible for the core coding tasks and integrating different parts of the codebase), *Application Designers* (manages data collection, chooses appropriate data structures, and ensures data accuracy and application efficiency), and Documentation Specialist (handles the project report, documentation, and presentation materials).
   Note: *Though each member has a specific role, all the members should know the project internals like code structure, data structures, algorithms used, logic and control flow, error/exception handling, performance or time complexity etc.*
4) For proper group collaboration, use a group chat tool (e.g., WhatsApp) for daily communication and updates. Use Git for version control. Create a repository on a platform like GitHub or GitLab. Ensure all members are familiar with basic Git commands.
5) Given the time of around 15 days for the task, schedule at least a check-in meeting once every 3 days to discuss progress, address issues, and adjust plans as necessary. Define key milestones for the project, such as completing the core functionality, optimizing performance, and finalizing the CLI interface etc., spawned across the given timeline.
6) The group lead is advised to break down the project into smaller tasks and allocate them to group members based on their roles and strengths. Encourage pair programming for complex tasks to ensure knowledge sharing and improve code quality. Implement regular code reviews to ensure code quality and consistency. Rotate reviewers to get different perspectives.

7) **Project Deliverables:**

Each group (only the group lead) should submit the following by the due date via Canvas.

- ✓ The complete Java codebase as a zip file
- ✓ A detailed project report in PDF format covering the following.
  - project overview,
  - objectives,
  - design and implementation details (code structure, description of classes),
  - data structures and algorithms used with justification
  - performance (time complexity) analysis and optimization techniques,
  - challenges faced and solutions implemented,
  - instructions for using the application (with sample screenshots)
- ✓ The project presentation slides in PDF format to demonstrate your application. The slides may include the following.
  - Project objectives and chosen application scenario.
  - Key features and functionalities.
  - Demonstration of the CLI commands/GUI features and outputs.
  - Team contributions and roles.

SUBMISSION DEADLINE: 2nd August 2024, 11:55 PM

GRADING CRITERIA: As mentioned in the syllabus document, this group project is worth **15%** of the overall course grade.

Your project will be evaluated based on the following criteria:

1) Functionality (**40 points**):
   a. Core functionality as per the chosen application scenario (20 points)
   b. Correct implementation/use of data structures and algorithms (10 points)
   c. Proper handling and processing of data (10 points)
2) Code Quality (**20 points**):
   a. Code readability and organization (10 points)
   b. Appropriate use of comments and documentation (5 points)
   c. Adherence to Java best practices and coding standards (5 points)
3) Application Performance (**15 points**):
   a. Efficiency of implemented algorithms (10 points)
   b. Ability to handle large datasets without performance degradation (5 points)
4) Interface (CLI/GUI) Design and Output (**15 points**):
   a. Intuitive and user-friendly interface (can be CLI or GUI) (10 points)
   b. Clear and well-structured output (5 points)
5) Documentation and Presentation (**10 points**):
   a. Comprehensive report detailing the design, implementation, and application usage (5 points)
   b. Presentation document, demonstrating work, the application and its features (5 points)

Note: The above-mentioned components will be examined and evaluated based on the submission as well as the oral presentation. The presentation schedule will be announced in due course of time.

APPLICATION SCENARIOS (Climate Change)

1) **Climate Data Analysis and Visualization**: Develop an application that analyzes and visualizes climate data (e.g., temperature, precipitation, CO_2 levels) from different regions of Ghana/Africa over time. Your application should allow users to load data, sort and search data, and generate summaries. The key features of your application can be as follows.
   Data Loading: Command to load climate data from a file.
   Data Analysis: Commands to sort data by various attributes (e.g., year, region) and search for specific data points.
   Data Visualization: Command to generate text-based summaries.
   CLI Commands/Menu:
   load_data <filename>
   show_summary
   sort_data <attribute>
   search_data <attribute> <value>
   generate_chart <attribute>

2) **Wildfire Simulation and Prediction**: Create a simulation to model and predict the spread of wildfires based on factors like temperature, humidity, wind speed, and vegetation. The application should allow users to set parameters, run simulations, and visualize the spread of wildfires over time. The key features of your application can be as follows.
   Simulation Initialization: Command to set up the simulation environment.
   Parameter Setting: Commands to input environmental factors.
   Simulation Execution: Command to run the simulation.
   Visualization: Command to display fire spread using text-based grids.
   CLI Commands/Menu:
   init_simulation <grid_size>
   set_parameters <temperature> <humidity> <wind_speed>
   run_simulation <iterations>
   show_fire_spread

3) **Climate Impact on Biodiversity:** Design an application to study the impact of climate change on different species and ecosystems. The application should allow users to load species data, analyze interactions, and generate impact reports. The key features of your application can be as follows.
   Data Loading: Command to load species and ecosystem data.
   Impact Analysis: Command to analyze the impact of climate factors on biodiversity.
   Report Generation: Command to generate and display impact reports.
   Search Functionality: Command to search for specific species or interactions.
   CLI Commands/Menu:
   load_species_data <filename>
   analyze_ecosystem <ecosystem>
   show_impact
   find_species <species_name>

4) **Climate Change News Aggregator**: Develop an application to aggregate and analyze news articles related to climate change from various sources. The application should allow users to add articles, search by keywords, sort by date, and display summaries. The key features of your application can be as follows.
   Article Addition: Command to add news articles.
   Keyword Search: Command to search articles by keywords.
   Sorting: Command to sort articles by date.
   Summary Display: Command to display article summaries.
      CLI Commands/Menu:
      add_article <title> <content>
      search_keyword <keyword>
      sort_by_date
      display_summary

5) **Climate Change Disaster Response Management**: Create a system to manage and optimize disaster response efforts related to climate change events such as hurricanes, floods, and wildfires. The application should allow users to input disaster data, allocate resources, prioritize tasks, and display summaries.
   Disaster Data Input: Command to input data for climate change-related disasters.
   Resource Allocation: Command to allocate resources for disaster response.
   Task Prioritization: Command to prioritize response tasks.
   Summary Display: Command to display response summaries.
   CLI Commands/Menu:
      input_disaster_data <disaster_type> <location>
      allocate_resources <resource> <location>
      prioritize_tasks
      show_summary

APPLICATION SCENARIOS (General)

6) **Virtual File System Organizer:** Your project will simulate a virtual file system organizer to manage a collection of virtual files and directories. The application will allow users to create, delete, move, and search for files or directories within the system. It will not involve any real file manipulation on the disk but will instead use in-memory data structures to simulate these operations. The key features of your application can be as follows:
   File and Directory Management:
      a. Create Files/Directories: Commands to create new files and directories.
      b. Delete Files/Directories: Commands to delete existing files and directories.
      c. Move Files/Directories: Commands to move files and directories within the virtual file system.
   Search and Organize:
      d. Search: Commands to search for files and directories by name or other attributes.

e. Sort: Commands to sort files and directories based on attributes like name, type, or creation date.

Directory Structure Visualization:

f. Show Structure: Command to display the current structure of directories and files in a tree format.

CLI Commands/Menu:

create_file <file_path>: Create a new file at the specified path.
create_dir <dir_path>: Create a new directory at the specified path.
delete <path>: Delete the file or directory at the specified path.
move <source_path> <destination_path>: Move a file or directory from the source path to the destination path.
search <attribute> <value>: Search for files or directories based on a specified attribute and value.
sort <attribute>: Sort files and directories by the specified attribute.
show_structure: Display the current directory and file structure in a tree format.

7) **Event Scheduler and Calendar:** This project allows users to create, modify, delete, and view events within a virtual calendar. The application will not require any persistent storage and will store all data in memory using appropriate data structures. The key features of your application can be as follows:

Event Management:

- Create Event: Command to add new events to the calendar.
- Modify Event: Command to update details of existing events.
- Delete Event: Command to remove events from the calendar.
- View Events: Command to display events based on various filters (e.g., date, month).

Event Searching and Sorting:

- Search Events: Commands to search for events by attributes like title, date, or location.
- Sort Events: Commands to sort events based on attributes such as date, title, or priority.

Event Summaries:

- Generate Summaries: Command to generate text-based summaries of events for a specified date or range of dates.

CLI Commands/Menu:

create_event <title> <date> <time> <location> <description>: Create a new event with specified details.
modify_event <event_id> <attribute> <new_value>: Modify an existing event's attribute with a new value.
delete_event <event_id>: Delete the event with the specified event ID.
view_events <filter>: View events based on the specified filter (e.g., date, week, month).
search_event <attribute> <value>: Search for events based on a specified attribute and value.

sort_events <attribute>: Sort events by the specified attribute.
generate_summary <date_range>: Generate a summary of events for the specified date range.

8) **Interactive Restaurant Reservation System:** This project involves creating a simulation of a restaurant reservation system that allows users to book tables, cancel reservations, and view reservation statuses. The system will operate entirely in memory without the need for persistent storage, relying on data structures to manage the state and operations. The key features of your application can be as follows:

Reservation Management:

- Book Table: Command to book a table for a specified date and time.
- Cancel Reservation: Command to cancel an existing reservation.
- View Reservations: Command to view all reservations or filter by date, time, or customer name.

Table Availability:

- Check Availability: Command to check table availability for a specified date and time.

Reservation Summaries:

- Generate Summaries: Command to generate text-based summaries of reservations for a specified date or range of dates.

CLI Commands/Menu:

book_table <customer_name> <date> <time> <num_people>: Book a table for a specified customer on a given date and time for a certain number of people.
cancel_reservation <reservation_id>: Cancel the reservation with the specified reservation ID.
view_reservations <filter>: View reservations based on the specified filter (e.g., date, customer name).
check_availability <date> <time>: Check table availability for a specified date and time.
generate_summary <date_range>: Generate a summary of reservations for the specified date range.

9) **Traffic Simulation System:** You will create a simulation of a traffic system to manage vehicle flow at a four-way intersection. The system will simulate traffic lights, vehicle queues, and the logic that determines when vehicles pass through the intersection. The project will not require persistent storage and will use in-memory data structures. The key features of your application can be as follows:

Traffic Light Management:

- Simulate Traffic Lights: Commands to control traffic lights at the intersection (e.g., green, yellow, red).
- Light Timing: Commands to set the duration of each light (e.g., green for 30 seconds).

Vehicle Queue Management:

- Add Vehicles to Queue: Command to add vehicles to the queue at each intersection.

- Remove Vehicles from Queue: Command to remove vehicles from the queue as they pass through the intersection.

Intersection Logic:

- Traffic Flow Logic: Implement logic to determine which queue moves based on the traffic light state and the order of vehicles in the queue.
- Emergency Vehicle Logic: Optional: Implement priority logic for emergency vehicles.

Simulation Control:

- Start/Stop Simulation: Command to start or stop the traffic simulation.
- Simulation Speed: Commands to adjust the speed of the simulation (e.g., real-time, fast-forward).

CLI Commands/Menu:

set_light <direction> <color>: Set the traffic light color (green, yellow, red) for a specified direction (e.g., north, south, east, west).

set_light_timing <color> <duration>: Set the duration of a specified light color (e.g., green for 30 seconds).

add_vehicle <direction> <vehicle_type>: Add a vehicle to the queue in a specified direction (e.g., car, truck, bus).

remove_vehicle <direction>: Remove a vehicle from the queue in a specified direction.

start_simulation: Start the traffic simulation.

stop_simulation: Stop the traffic simulation.

set_simulation_speed <speed>: Set the speed of the simulation (e.g., real-time, fast-forward).

view_queue <direction>: View the current queue of vehicles in a specified direction.

GOOD LUCK WITH YOUR LEARNING!!!