**General Assignment Policies for Java Programming**

**Submission Requirements**

1. **Submission Deadlines:**
   * Submit by 31st July.
   * Penalty for late submissions: 10% per day, up to 2 days. No submissions accepted beyond 2 days without prior arrangements.
2. **Plagiarism:**
   * Original work required. Collaboration on concepts is okay, but implementation must be individual. Plagiarism will result in a zero and potential disciplinary action.
3. **Code Quality:**
   * Code should be clean, well-documented, and follow naming conventions and coding standards.
4. **Testing:**
   * Provide comprehensive test cases with test inputs and expected outputs.
5. **Documentation:**
   * Include approach, pseudocode, code explanation, assumptions, limitations, and a user manual if necessary.

**Grading Rubrics**

1. **Code Functionality (30%):**
   * Excellent: Meets all requirements, handles edge cases/errors.
   * Good: Meets most requirements, handles most cases/errors.
   * Satisfactory: Meets basic requirements, handles some cases/errors.
   * Needs Improvement: Significant errors, handles few edge cases.
2. **Code Quality (20%):**
   * Excellent: Clean, well-organized, follows standards.
   * Good: Mostly clean, minor deviations.
   * Satisfactory: Somewhat organized, several deviations.
   * Needs Improvement: Disorganized, doesn’t follow standards.
3. **Documentation (20%):**
   * Excellent: Comprehensive, covers all aspects.
   * Good: Clear, covers most aspects.
   * Satisfactory: Basic, covers some aspects.
   * Needs Improvement: Incomplete or unclear.
4. **Testing (20%):**
   * Excellent: Comprehensive test cases covering a wide range of inputs.
   * Good: Several test cases, covers some inputs.
   * Satisfactory: Basic test cases for limited inputs.
   * Needs Improvement: Few or no test cases.
5. **User Interface (10%) (if applicable):**
   * Excellent: Intuitive, user-friendly, fully functional.
   * Good: Mostly intuitive, minor issues.
   * Satisfactory: Functional, significant usability issues.
   * Needs Improvement: Difficult to use or non-functional.

**Project Tasks**

1. **Data Collection and Modeling:**
   * Define data structures for real-time traffic data (e.g., vehicle counts, speeds).
   * Consider how data from different intersections will be collected and processed.
2. **Algorithm Design:**
   * Develop algorithms to dynamically adjust traffic signal timings.
   * Consider factors like traffic density, vehicle queues, peak hours, and pedestrian crossings.
3. **Implementation:**
   * Create a Java application to integrate with traffic sensors and control signals.
   * Ensure real-time adjustment of signal timings based on traffic conditions.
4. **Visualization and Reporting:**
   * Develop visualizations for real-time traffic monitoring.
   * Create reports on traffic flow, wait times, and congestion reduction.
5. **User Interaction:**
   * Design a user interface for traffic managers to monitor and adjust signal timings.
   * Develop a dashboard for city officials to view performance metrics and historical data.

**Deliverables**

* **Data Flow Diagram:** Shows data collection, analysis, and optimization processes.
* **Pseudocode and Implementation:** Detailed pseudocode and Java code for traffic signal algorithms.
* **Documentation:** Design decisions, data structures, assumptions, and potential improvements.
* **User Interface:** Interfaces for traffic managers and city officials.
* **Testing:** Comprehensive test cases validating system functionality.