**CSADPRG MCO3**
**Specifications for Comparative Analysis of Programming Languages**

## 1. Activity Description

This activity builds on the theoretical foundations of programming languages covered in class by examining and comparing the features and constructs of selected languages. For this iteration of the MCO, each group will study Go, Kotlin, R, and Ruby. The project consists of three components: a language evaluation paper, software development in multiple programming languages, and a class presentation. By completing this project, students will develop an understanding of the paradigms supported by each language and gain insight into their similarities and differences.

## 2. Evaluation Paper

The language evaluation paper should provide a detailed discussion of the features and constructs of Ruby, R, Kotlin, and Go. Using code snippets from two programming exercises (i.e., the load calculator and corpus analysis), the paper should evaluate each language about the theories and concepts discussed in class. A section should explain how specific language features supported or impeded the implementation of these exercises. Required content varies by language, depending on the relevant programming paradigm, and the outline is provided in Section 4.

**Formatting Requirements:**

- **Margins:** 1 inch on all sides
- **Font:** 12pt Times New Roman for text; 11pt Bold Courier New for code snippets
- **Paper Size:** 8"x11" (Short)
- **Spacing:** Single (except between paragraphs)

**Important:**
Any group with "copy-pasted" material from external sources will automatically receive a grade of 0 for the project. Paper content should reflect your group's specific application and analysis. Concepts from reference materials may be paraphrased but must be appropriately cited. Papers without citations will incur deduction points.

**Citation Style:**

Use the American Psychological Association (APA) format. In APA, citations follow the author-date method, with the author's last name, the year of publication appearing in the text, and full references in the reference list.

**Example Citations:**

- Smith (1970) compared reaction times...
- In a recent study of reaction times (Smith, 1970)...

- In 1970, Smith compared reaction times...

**Evaluation Criteria:**

- **Content and Analysis (30 points)**
  - **Language Feature Analysis (10 points):** Depth of analysis on syntax, semantics, control flow, data structures, etc.
  - **Comparative Analysis (10 points):** Effectiveness in comparing and contrasting languages; identification of strengths, weaknesses, and unique features; clear implications for software development.
  - **Code Snippet Analysis (10 points):** Insightful interpretation of code snippets, identifying language-specific features and best practices, and evaluating code efficiency, readability, and maintainability.
- **Writing and Organization (15 points)**
  - **Clarity and Coherence (5 points):** Clear organization of ideas with effective transitions; concise, clear writing.
  - **Grammar and Mechanics (5 points):** Correct grammar, punctuation, and spelling; adherence to APA style.
  - **Citations and References (5 points):** Accurate, consistent citations and a well-formatted reference list.

## 3. Class Presentation

Each group will present their software application and language analysis to the class in a 20-minute presentation. This presentation provides an opportunity to share insights on the programming languages and paradigms studied.

**Presentation Requirements:**

- **Time Limit:** Adhere strictly to the 20-minute time limit.
- **Slide Deck Requirements:**
  - Clear, consistent design
  - Maximum of 15 slides
  - Minimum font size of 24 points
  - Minimize text and use visuals for clarity
- **Audience Engagement:**
  - Use interactive elements, such as live coding demos or quizzes
  - Maintain eye contact and speak clearly
- **Q&A Session:**
  - Allocate 5 minutes for Q&A
  - Respond to questions concisely

**Presentation Content:**

- Overview of the programming paradigm
- Brief history of the language

- Software demonstration
- Source code discussion, highlighting essential features and capabilities (only those implementing data structures and algorithms must be presented)

## 4. Project Schedule and Deliverables

- **Language Evaluation Paper Due:** November 25, 2024
- **Class Presentation:** November 25-29, 2024
- **Slide Submission Deadline:** November 29, 2024

## 5. Paper Format

Submit the evaluation paper on Canvas by 11:59 PM. Late submissions will not be accepted, and a grade of 0 will be given for late papers. The evaluation paper should include:

- **Title Page:**

*An Evaluation Paper on <Programming Languages>*
Submitted in partial fulfillment of CSADPRG requirements
[Group member names in alphabetical order]
[Instructor's Name]
[Date]

- **Sections:**
  1. **Introduction:** Overview of the programming paradigms and languages studied; discuss language history, revisions, current state, and application domains.
  2. **Language Comparison:** Major features of each language; explain how language features (positively and negatively) affected the development of the 2 applications with code snippets  You may have separate subsections sections foir each application.
  3. **Conclusion:** Summary of findings, insights on language suitability for your application, and recommendations.
  4. **References:** Follow APA format.

## 6. Additional Guidelines

- **Academic Integrity:**
  - All work must be original and cited appropriately.
  - Plagiarism will result in a failing grade.
- **Group Collaboration:**
  - Clearly define roles and responsibilities for each group member.
  - Ensure equal contributions from all members.
- **Late Submission Policy:**
  - Late submissions will be penalized.