CMSC 330: Programming Languages

Research Paper and Presentation Assignment

Overview

This assignment is intended to deepen your understanding of *programming languages as a discipline*. You will study a topic that is conceptually grounded in CMSC 330, engage with scholarly and technical literature, and produce both a formal research paper and a professional oral presentation.

The emphasis is on programming language concepts such as language design, paradigms, abstraction mechanisms, semantics, type systems, runtime models, and the impact of modern tooling (including AI-assisted programming) on how languages are used and understood.

Deliverables and Points

- **Research Paper**: 100 points
- **Presentation**: 100 points

There is **no proposal phase**. You are responsible for selecting an appropriate topic and managing scope.

Acceptable Topics and How to Approach Them

Your topic **must be clearly related to programming languages** and accessible at an undergraduate level. You are *not* expected to invent new theory. Instead, you should demonstrate that you can explain, analyze, and evaluate programming language concepts using ideas from CMSC 330.

To make expectations concrete, you are recommended to structure your topic around **one of the following practical approaches**.

Option A: Single Programming Language Focus

Pick **one programming language** and organize your paper as follows:

1. **Brief introduction and motivation**
    - Why this language exists
    - What problems it was designed to address

    - Typical application domains
2. **Core language features** (focus on CMSC 330 concepts)
    - Paradigm(s) supported (functional, object-oriented, etc.)
    - Type system (static vs dynamic, inference, safety)
    - Abstraction mechanisms (functions, classes, modules, pattern matching)
3. **Runtime and semantics-related aspects**
    - Execution model (compiled, interpreted, virtual machine)
    - Memory management strategy
    - Evaluation strategy (eager, lazy, hybrid)
4. **Critical evaluation**
    - Use language evaluation criteria discussed in class (readability, writability, reliability, cost)
    - Strengths and limitations
5. **Modern relevance**
    - How the language interacts with modern tooling (IDEs, compilers, AI-assisted programming)

Examples: Python, Rust, Java, OCaml, JavaScript, Go, Haskell

Option B: Comparison of a Small Language Family

Compare **two or three closely related languages**.

Suggested structure:

- Shared design goals
- Key differences in syntax, typing, or semantics
- Trade-offs introduced by these differences
- Evaluation using course criteria

Examples:

- C vs. Rust
- Java vs. Kotlin
- Python vs. JavaScript
- ML vs. Haskell

Option C: Focused Language Feature or Mechanism

Study **one programming language feature**, grounded in concrete languages.

Suggested structure:

- What the feature is and why it matters
- How it is implemented or expressed in real languages
- Benefits and drawbacks
- Impact on correctness, readability, or performance

Examples:

- Garbage collection strategies
- Pattern matching
- Static vs dynamic typing
- Concurrency models (threads, async/await, actors)

Note on AI-Assisted Programming and "Vibe Coding"

AI-assisted programming (sometimes informally called *vibe coding*) may be treated as a **language-like abstraction layer** rather than a traditional programming language.

If you choose this topic, your paper should:

- Clearly explain *why* it can be viewed as language-like (syntax, semantics, translation to executable code)
- Analyze how it affects readability, correctness, and programmer understanding
- Ground the discussion in concrete languages that the AI ultimately targets (e.g., Python, Java)

Pure opinion pieces about AI are **not acceptable**; the analysis must remain anchored in programming language concepts.

The Research Paper (100 points)

Length and Format

- Minimum **5 full pages** of content (not including references)
- Double-spaced
- 12-point Times New Roman (or equivalent)
- 1-inch margins
- Submitted as a **PDF**

First Page (Journal-Style Front Matter)

The first page should resemble the opening page of a simplified academic journal article. You must follow the formatting instructions below **exactly**.

Page Setup (Microsoft Word)

- Same margins as the rest of the paper (1 inch on all sides)
- No page number on the first page

Title

- Centered horizontally
- **Bold**
- Font: Times New Roman
- Font size: **16 pt**
- Title Case capitalization

Leave **two blank lines** after the title.

Author and Affiliation Block

Center the following block on the page:

- Student Name (12 pt)
- CMSC 330: Programming Languages (12 pt)
- Institution name (12 pt)
- Email address (12 pt)

All text:

- Times New Roman
- Single-spaced within the block
- One blank line between each item

Leave **two blank lines** after the affiliation block.

Abstract

- Heading **Abstract** centered and bold (12 pt)
- One blank line after the heading
- Abstract text:
    - 150–200 words
    - Single paragraph
    - Fully justified or left-aligned
    - Times New Roman, 12 pt

The abstract should briefly state:

- The topic and scope of the paper
- The main programming language concepts discussed
- The perspective or evaluation you will present

Leave **two blank lines** after the abstract, then begin the main body of the paper (double-spaced).

Content Requirements

- At least **5 technical references**, with **at least 3 from the ACM Digital Library**
- References must use **ACM citation format** with in-text citations such as [1], [2,3]
- The paper must synthesize sources into a **coherent argument or analysis**, not a sequence of paper summaries
- Clear organization with logical transitions between sections

If your topic focuses on one or more programming languages, you must include a **critical evaluation using language evaluation criteria** discussed in the course (for example, readability, writability, reliability, cost, and abstraction support).

Discussion of broader implications (such as correctness, maintainability, or ethical concerns related to language use or tooling) is appropriate **only insofar as it supports the technical analysis**.

Evaluation Criteria

You will be evaluated on:

- Technical depth and correctness
- Understanding of programming language concepts
- Quality of analysis and synthesis
- Organization and clarity
- Writing mechanics and citation accuracy

The Presentation (100 points)

- Length: **approximately 10 minutes**
- Significant deviations from the time limit will result in deductions
- Must be supported by presentation media (e.g., slides)

Your presentation should:

- Clearly motivate the topic
- Explain key programming language concepts involved

- Present analysis or conclusions, not just background
- Be understandable to classmates who have completed CMSC 330

Grading Rubric: Research Paper (Writing)

Each category is scored on a 0–4 scale.

1. Central Idea and Technical Focus

- **4**: Clear, original, and technically focused thesis grounded in programming languages
- **3**: Clear and appropriate thesis with solid technical relevance
- **2**: General or weakly focused thesis; limited technical depth
- **1**: Unclear or poorly defined focus
- **0**: No identifiable central idea

2. Organization and Coherence

- **4**: Logical, well-structured argument with strong transitions
- **3**: Clear organization with mostly effective transitions
- **2**: Basic organization; transitions are weak or mechanical
- **1**: Difficult to follow; poor structural planning
- **0**: No discernible organization

3. Technical Development and Use of Sources

- **4**: Sophisticated synthesis of high-quality sources; strong technical analysis
- **3**: Appropriate sources with good explanation and integration
- **2**: Limited synthesis; sources used mostly descriptively
- **1**: Minimal or poorly explained use of sources
- **0**: No credible sources or citations

4. Mechanics and Citation Style

- **4**: Virtually no grammatical or citation errors; correct ACM format
- **3**: Minor errors that do not impede understanding
- **2**: Noticeable errors; inconsistent citation formatting
- **1**: Frequent errors that distract from content
- **0**: Pervasive mechanical or citation failures

5. Style and Professional Tone

- **4**: Clear, precise, and professional academic style
- **3**: Generally clear and appropriate tone

- **2**: Adequate but uneven or informal in places
- **1**: Inappropriate or confusing style
- **0**: Style seriously undermines comprehension

Grading Rubric: Presentation (Speaking)

Each category is scored on a 0–4 scale.

1. Content Accuracy and Depth

- **4**: Accurate, insightful explanation of programming language concepts
- **3**: Correct and clear explanation with minor omissions
- **2**: Basic understanding; limited depth
- **1**: Significant misunderstandings
- **0**: Content is incorrect or incoherent

2. Organization and Clarity

- **4**: Well-structured, easy to follow, strong narrative flow
- **3**: Clear structure with minor issues
- **2**: Some organization, but transitions are weak
- **1**: Hard to follow
- **0**: No apparent structure

3. Delivery and Speaking Skills

- **4**: Confident, clear, professional delivery
- **3**: Generally clear with minor delivery issues
- **2**: Uneven delivery; reliance on notes
- **1**: Difficult to hear or understand
- **0**: Presentation not delivered

4. Visual Aids

- **4**: Slides are clear, well-designed, and support the talk
- **3**: Slides are helpful but could be improved
- **2**: Slides are cluttered or minimally helpful
- **1**: Slides distract from content
- **0**: No visual support

5. Time Management and Engagement

- **4**: Within time limits; engages audience effectively

- **3**: Minor timing issues; reasonable engagement
- **2**: Noticeable timing problems or limited engagement
- **1**: Major timing issues; little engagement
- **0**: Presentation not completed