Prompting, Debugging, and Innovation for Code Generation with LLMs

*This is an **individual assignment**. The final submission must be completed and submitted independently by each student.*

***Deadline: Oct 22, 11:59PM EST**. The deadline is sharp -- late submissions will not be accepted. Assignments must be submitted through GradeScope. We understand that circumstances may arise where you may need to submit the assignment late. **Seek approval by contacting TA Mingzhe Li (mingzhel@umass.edu) at least 24 hours in advance (unless it's a last-minute emergency and you cannot).** Medical conditions, religious or funerary events, university-related events (conference visit, athletic event, field trip, or performance), or extenuating non-academic reasons (military obligation, family illness, jury duty, automobile collision) that need extension will be accommodated with written documentation. Assignments or exams from other courses, interviews, or paper deadlines are not legitimate reasons for an extension.*

Objective: Students will explore how Large Language Models (LLMs) generate code, experiment with prompting strategies, practice debugging, and propose new prompting or workflow solutions. (Total: 20 points)

LLM Selection Requirement:
For all parts of this assignment, students must use at least two LLMs from different model families. Examples of distinct families include GPT (e.g., GPT-4, GPT-5), LLaMA (e.g., LLaMA-2, LLaMA-3), Vicuna, Qwen, Claude (Anthropic), and Gemini (Google DeepMind). For instance, GPT-4 and GPT-5 belong to the same family. Evaluations conducted only within a single family may lack generalizability.

Problem Selection Requirement: You may either:
1. Select problems from existing datasets:
   - HumanEval https://github.com/openai/human-eval
   - HumanEval+ https://huggingface.co/datasets/evalplus/humanevalplus

- APPS	https://huggingface.co/datasets/codeparrot/apps
- HumanEval-ET	https://huggingface.co/datasets/dz1/CodeScore-HumanEval-ET
- APPS+	https://github.com/Ablustrund/APPS_Plus
- MBPP-ET	https://huggingface.co/datasets/dz1/CodeScore-MBPP-ET
- SWE-Bench	https://github.com/SWE-bench/SWE-bench

2. Write your own programming problems. If you choose this option, you must provide natural language description and write comprehensive test cases for each problem. Comprehensive test cases means:

- Covering the happy path: typical valid inputs where the program is expected to work normally.
- Covering exceptional paths: edge cases, boundary conditions, invalid inputs, and error scenarios where robust code should still behave predictably.

Part 1: Prompt Design & Code Generation (40% – 8 points)
Select 10 programming problems (following the requirement above). Apply at least two prompting strategies given below. For each problem, evaluate correctness with the pass@k metric across at least two LLMs from different families. Include the exact prompt(s) you used in your report. Summarize results in a clear table with table explanation and discussion if any.

- Chain-of-Thought (CoT)
- Stepwise Chain-of-Thought (SCoT)
- Self-Planning
- Self-Debugging
- Self-Edit
- Self-Repair

Part 2: Debugging & Iterative Improvement (30% — 6 points)
From the 10 problems in Part 1, identify at least two failure cases where the generated code failed test cases. If all 10 problems pass, you must select additional problems until you can analyze at least 2 failure problems. For each failure case:

- Use test cases to identify the failure.
- Refine prompts or provide debugging hints to improve performance.
- Document what changes you made, what worked, and what didn't.
- Analyze why the model struggled (e.g., reasoning gap, poor error handling, misunderstanding of problem).
- Compare how different LLM families fail on the same problem and whether their debugging improvements differ in effectiveness.

– Include the exact prompt(s) for both the failed attempt and the improved attempt in your report.

Part 3: Innovation — Propose Your Own Strategy (30% — 6 points)

- Propose and test a novel strategy to improve LLM code generation.
  - This may involve new prompts, workflows (multi-step generation → refinement → testing), role-based prompting, tool integration, or hybrid approaches.
- Report whether it improves results.
- If your idea fails, you must analyze why (what assumptions were wrong, what could be done differently).
- Apply your proposed strategy to at least two LLMs from different families and analyze its effect across models.
- Include any prompts, workflow descriptions, or scripts you used in your report.

Deliverables:

Submit one PDF document on the course platform. The PDF should include:

- Prompts, methodology, experiments, results (with pass@k), debugging analysis, and innovation discussion.
- A link to your GitHub repository, which must contain:
  - Prompts and/or workflows/scripts used
  - Generated code
  - Test cases (dataset-provided or self-written)
  - Evaluation scripts and results