Overview. The goal of this project is to dive deeper into language design and implementation by developing a significant extension to MyPL. The project will largely be graded on: (1) the ***quality of your work***; (2) the ***completeness of your work***; (3) the ***quality of your tests and documentation***; and (4) the ***difficulty of your project***. The project is worth 80 points and includes a proposal, a project specification, and a demo presentation (video and slides).

Step 1 (due by Monday, March 18th). The first step is to identify *two* extensions you are interested in exploring.[1] The following is a non-exhaustive list of project ideas (by general difficulty level). You are not required to select one of the following—each is just an idea of a possible project that you may be interested in or may help give you ideas in terms of picking your own direction. You should also look at existing program languages for inspiration.

Significantly Difficult: These projects are more involved and generally require additional thinking about design decisions as well as some independent research into how each would work. Note these would score significantly higher on the "level of difficulty" project criteria.

- Concurrency/multi-threading support.
- Compilation to another interpreter (e.g., JVM or CPython bytecode, WASM, LLVM IR).
- A binary-based bytecode for our VM (extending what we've done in class)
- A completely separate language syntax design that "compiles" to our VM
- Compilation to assembly language (e.g., to ARM or x86).
- Fully functional implementation of a garbage collector.
- Full featured module/import system (e.g., like "import" in Python and Java)

More Challenging: These projects are less challenging than those above, but still require thought about design issues and/or involve some implementation challenges. These projects would generally score on the higher end of the "level of difficulty" evaluation criteria.

- Classes (with public/private member variables and methods).
- Pointers and/or pass by reference.
- Function overloading.
- Operator overloading.
- Struct type subclassing (inheritance).

[1] Because each student in the class will need to work on a different extension, you must provide a first and a second choice. Assuming neither of your choices have already been assigned, you will receive your first choice, followed by your second choice (if the first choice is taken), or else be notified to select an additional alternative.

- Performance optimizations (pick a few optimizations, implement and benchmark them).
- Higher-order functions and closures.
- Transcompilation to another language (e.g., C, C++, or Python).

Less Challenging: These projects are more straightforward and are generally easier to implement than those above. These projects would generally score on the mid-to-low end of the "level of difficulty" evaluation criteria.

- Additional collection types (e.g., dictionaries, lists, data frames).
- List comprehensions.
- Exception handling (e.g., try-catch blocks).
- Additional "common" constructs (e.g., switch statements, for each loops).
- Constant variables and function args.
- Proper handling of associativity and precedence.
- Collections of new operations and operators.
- File I/O.
- Delete operation.

Tooling Projects: These projects (which are also acceptable) involve adding "tool" support to MyPL as opposed to language extensions. The level of difficulty differs and depends on the project specifics.

- A debugger (e.g., with checkpoints, stepping capability, variable inspection).
- A fully functional REPL (read-eval print loop) for MyPL.
- A memory or execution profiler (execution monitoring/statistics).
- Save and read MyPL programs in our IR (to "compile" to a file, then execute later).

Again, you are free to choose something different than what is listed above. The difficulty level of your project will generally be based on the scope of your project, the degree of modifications and/or additions needed to implement the extension, the complexity of the extension itself, the amount of research you had to do to implement the project, and the amount of testing and quality assurance you perform.

Submitting your Choices. You must submit your first and second choice preferences as a hard copy in class on or before the due date. For your first and second choices, you must state each extension and provide a short description of what you plan to tackle as part of the extension (the scope). This description should contain a list of the "features" of the extension (e.g., what a user would be able

to do, what capabilities will be added to MyPL, etc.). Note that these are just your initial thoughts to help me understand what you are thinking in terms of the project. Note that a special "project" repository will be made available in GitHub for you to store and submit your project source code and documentation.

Step 2 (due by Friday, April 5th). For this step, you must hand in a hard-copy specification (plan of attack) for your extension and what you have already accomplished in class. The details of the specification will differ depending on your project. At a minimum, the specification should include the scope of your extension, a detailed plan for how you will test your extension, an initial design (e.g., what will need to be modified in the MyPL implementation, what will need to be added, etc.), and a weekly plan with milestones for completing the project. Note that you will have approximately five weeks to finish the project from this point.

Step 3 (due by Friday, April 26th). For this step, you must hand in a hard-copy status report of your progress. Your status report must provide details on what you have completed including concrete tests you will use for your extension (both unit tests and test programs). You should plan to have at a minium 10+ test programs and a large suite of unit tests for your extension. For this step, your project implementation should be ***close to 80% finished***.

Step 4 (due by the end of Friday of finals week.) Your project must be fully completed on or before the Friday of finals week. All of your source code must be pushed to your repository as well as the slides you use as part of your final project presentation. Your project presentation must consist of a "narrated" video that consists of:

- Your name, the course, the semester, and a brief desscription of the extension you implemented with examples.
- A demo of the extension "in action", in which you must: show how the extension would be used in a "real world" setting, give a walk-through of the (non unit) tests you developed, and show your extension working over the tests.
- A discussion of additional tests you developed, including the scope and examples of the unit tests you created.
- A description of the parts of the MyPL pipeline that you modified and at a high level what the modifications were.
- What was successfully completed, what if anything wasn't completed (compared to your specification in Step 2), and what you would do next if you had additional time.

Your video should be *at most* 12 minutes in length and must be made accessible either via zoom or youtube. Note that a large portion of your grade will be based on your demo. It is your responsibility to ensure I can access the video and that the video correctly presents the material. Finally, you must create a README.md file for your GitHub repository that, at a minimum, contains a link to your

video presentation and any additional instructions necessary to run your extension. All materials must also be available in your GitHub repository, including your slides, your test programs, your unit tests, and all source code needed to run your extension.

Grading Rubric

The following information describes how points will be allocated for your final project. Note that the project is worth a maximum of 80 points.

- **Steps 1–3 (10 points).** Steps 1–3 will each be worth a total of 2 points. An additional 4 points will be awarded at the end of the project if all three check-ins are turned in by their respective due dates.
- **Project Completeness (20 points).** You will receive up to 20 points based on whether and to what extent the project was fully completed. This includes whether all parts implemented work correctly and that all necessary parts were implemented. Note that if you have not fully implemented the base MyPL system (i.e., HW 1–6), points will still be deducted for not fully completing the extension if it relies on these base portions being finished.
- **Presentation and Slides (20 points).** You will receive up to 15 points for completeness and up to 5 points for the the quality of your presentation. Completeness here means that you covered all parts asked for with respect to your video presentation. The quality will be based on a 5-point scale, where 5 points implies excellent quality, 4 points implies satisfactory quality, and so on. Note that the points in this category are predicated on the successful completion of your project.
- **Difficulty of Extension (10 points).** You will receive up to 10 points based on the overall difficulty of the project. As mentioned above, the difficulty will be based on a number of factors. Similar to the presentation, the points in this category are predicated on successful conpletion of your project.
- **Overall Quality of Extension (10 points).** You will receive up to 10 points based on the overall quality of your implementation. Overall quality will be based on assement of the code you write (including comments, coding style, etc.) as well as from the demonstration of your extension as part of your presentation.
- **Scope and Quality of Testing (10 points).** You will receive up to 10 points for the scope and quality of your tests, which includes the overall coverage of your tests as well as explanations for each test (via comments and explanations provided in your presentation).