**Overview.** The goal of this project is to dive deeper into language design and implementation by developing a significant extension to MyPL. The project is broken into multiple steps, each of which have either a check-in or deliverable that you will need to turn in. The project will be graded on the *quality of your work*, the *completeness of your work*, the *quality of your tests and write ups*, and the *difficulty of your project*. The project is worth 10% of your final grade. Please be sure to ask if you have questions concerning the project requirements and/or expectations.

**Step 1 (due by Thur, March 25th).** The first step is to identify *two* extensions you are interested in exploring.[1] The following is a non-exhaustive list of things we've discussed in one form or another in class as possible projects.

- Language Extensions:
  - New built in types (e.g., lists, dictionaries, sets, arrays)
  - Classes (e.g., adding functions to types)
  - Type inheritance
  - Function overloading
  - Functional features (e.g., passing functions as arguments)
  - Exception handling (e.g., try-catch blocks)
  - Function type inference (for implicit function types)
  - Function and type prototypes
  - Constant variables and function args (for user-defined types)
  - Pointers and/or pass by reference
  - Suite of new operations and operators
  - Proper handling of associativity and precedence
  - Operator overloading
  - File I/O
  - Multithreading/concurrency support
  - Module system (for multifile programs)
  - A basic GUI framework (e.g., for graphics)
- Performance Extensions:
  - Garbage collector
  - A delete operation (to go along with new)
  - Translation to another language (e.g., C/C++ or Java) and benchmarking

[1] Because each student in the class will need to work on a different extension, I'm asking for your first and second choices. 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 by me to select an additional alternative.

  - Translation to bytecode (e.g., LLVM) and benchmarking
  - Performance optimizations and benchmarking

- Tool Extensions:
  - A REPL (read-eval print loop)
  - A debugger (e.g., with checkpoints, stepping capability, variable inspection)
  - A code editor (e.g., with syntax highlighting, syntax/type error detection)
  - A profiler (execution monitoring/statistics)
  - A unit test framework

- Additional Extensions:
  - A suite of built-in functions and example programs
  - A suite of "benchmark" programs and performance results
  - A suite of useful programs written in MyPL (may require additional built-in functions)

Note that you are free to choose something different than what is listed above. Also, the difficulty levels of the above vary considerably. An important part of developing your extension will be to scope out a reasonable (but challenging) project.

*Submitting your Choices.* You must submit your 1st and 2nd choice preferences using the form at https://forms.gle/huvPVyH1Lx7rnB2A7 on or before the due date. Note that extensions will be chosen based on the order of submission (so the sooner you submit the better). For your 1st and 2nd 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 basic description 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.

**Step 2 (due by Tues, April 6th).** For this step, you must submit a specification (plan of attack) for your extension. The details of the specification will differ depending on your project. At a minimum, the specification should include enough information to make it clear what the scope of the extension is and how it will work. You should think of the specification as a write up that we can use later to evaluate whether or not you completed your work. If you are doing a language extension, e.g., you would want to include the grammar for the new language constructs, basic examples of how the construct would work, and initial test cases you will use for testing. Instructions for submitting your specification will be provided via Piazza. Note that a special "Project" repository will made available for you via GitHub classroom to turn in your work.

**Step 3 (due by Tues, April 20th).** For this step, you will submit a progress update. By this point in the project, you should have a majority of the work completed. The progress update should detail what progress you have made and what you have left to finish. Additionally, any changes you need to make in terms of how your extension will work or the scope of the project should be

explained and justified. Your report will be submitted through your project GitHub repo. I will also look through the code you have submitted as well (i.e., pushed to the repo). You should provide instructions in your progress update so that I can build and demo your work as well.

**Step 4 (due by Tues, April 30th).** Your project must be completed on or before the Friday before finals week. All code must be pushed to your repository and a short write-up describing the extension, what was accomplished, what wasn't accomplished (from your progress report), the tests that you created, and instructions for building and running your work must be included. In addition, you will do a quick interview with me during finals week to demo your project and discuss challenges, etc. This interview will be scheduled the week before finals week.