CS 352 Final Project

Project proposal due Friday, October 22, 10pm Pacific
Complete submission due Friday, December 3, 10pm Pacific
In-class presentations November 30-December 6

The primary learning goal of this course is to give you theoretical tools and practical experience to learn new programming languages and language features more effectively. As a demonstration of this learning, you will design and complete a small project either in a programming language that has not been covered yet in your UP education, or using a language feature of an existing language that you have not yet studied.

The question you should keep in mind while designing and implementing this project is "*how is the programming language or framework you are using well or poorly suited to solve this problem?*" – this is the final project of a programming languages course, and while any problem can (in principle) be solved in any Turing-complete language, some are better choices than others for a given problem. The scope and complexity of the project should be in line with the other assignments in this course (approximately 100-300 lines of code, though this may vary by language; consult with the instructor if you need clarification), and should (much like the class assignments) demonstrate one or more features of the language or framework it is written in. Within those constraints, you have broad discretion to choose your own design.

You may, if you choose, work on this project with a partner. If you work together, your project should be approximately twice as complex as a solo project, and both partners will receive the same grade.

There are three main components to the project: an initial proposal (9% of your course grade, due in approximately two months), a project presentation (9% of your course grade, presented during the last week of class or the scheduled exam period), and the final submission (18% of your course grade, due the last day of classes).

PROJECT PROPOSAL

There are two parts to the project proposal: design document for the final project and a demo video of compilation of a "hello, world" code sample in your chosen language.

Design Proposal

The design proposal provides a general outline of your project idea, identifies the programming language the project will be written in, and states what language feature or features will be demonstrated, and how. This document is not a contract or specification of your final project, but rather the start of a conversation about its scope and direction. The proposal text should be at least 300 words, though should not exceed 600 words. If you work with a partner, your proposal should document at least two distinct language features your project will demonstrate;

the length of the proposal should be 450-900 words. If you have questions about project scope, or how to incorporate language features, please feel free to contact the instructor.

Demo Video

You should submit a short demo video demonstrating that you have installed the appropriate compiler or equivalent tools for your project, and can build and run a short program in your chosen language. You may use any tool you prefer to record this video; if you don't have an existing preference the instructor suggests TechSmith Capture, available through myapps.up.edu.

The code sample should also be submitted to Moodle. It does not need to do anything more complex than printing "hello, world" (or a language or framework-appropriate equivalent). The important part of the code sample is that it demonstrates you have investigated your chosen language in at least minimal depth.

Project Ideas

If you would like some ideas about a language to choose, here is a selection of currently popular languages. You may use a language not on this list instead if you prefer.

- Clojure *(a Lisp dialect (functional programming language) running on the Java VM)*
- Dart *(a language focused on UI and cross-platform client app development)*
- Elixir *(a functional programming language often used for concurrent programming)*
- Go *(a statically-compiled, garbage-collected language often used for networking)*
- Julia *(a language designed for high-performance scientific computing)*
- Kotlin *(a modern language that runs on the Java VM; used for Android development)*
- R *(a domain-specific language for statistics)*
- Ruby *(an object-oriented language often used for server-side web development)*
- Scala *(a JVM-based language combining functional and object-oriented paradigms)*
- Swift *(Apple's in-house language, primarily for iOS and Mac development)*
- TypeScript *(an extension of JavaScript with static typing information)*
- Zig *(an alternative to C for close-to-the-metal computation)*

If you would prefer to explore a language you have already learned in more depth, I would suggest investigating a library or framework that adds significant features to the language. Here are some examples that might be suitable for a project:

- Advanced uses of Haskell *(e.g. monad-based libraries)*
- C++ Guideline Support Library *(library-level memory safety features for C++)*
- Rust modules that add significant macro-based features *(e.g. Serde, a data serialization framework)*
- Various Python DSLs *(former students have demoed BeautifulSoup for HTML manipulation, the Django web framework, and sqlalchemy for database access)*

If there's a language on this list that interests you, but you don't have a good idea of what language features you want to highlight, you might try a web search for the release notes or new features of a recent language revision. You may also contact the instructor.

Submission

- Your submission should include a PDF document containing the following parts:
    - The design proposal, as described above.
    - Your preference for what day you wish to do your project proposal on.
        - If you find all of the days equally acceptable, please say so.
        - If not all student preferences can be accommodated the instructor will use a randomized method to decide whose are not.
        - Partners working together will share a presentation time slot.
    - Note that most document software has an "Export to PDF" option, and many operating systems include a PDF exporter in the list of standard printers. Consult the instructor if you are having technical difficulty producing a PDF.
- Your submission should also include the "hello, world" code sample.
    - If there are a large number of files or the directory structure is significant, this code may be placed in a .zip file.
- Your submission should include a video of you compiling and running the "hello, world" program
    - You may submit this either as a separate file, or a web link in the design proposal document
- All components should be submitted to the "Project Proposal" assignment dropbox on Moodle by the due date.
    - In the event your zipped code sample exceeds Moodle's 10 MB submission limit, include instructions for the instructor to access your code in your PDF document. A folder share through UP's Microsoft OneDrive service is likely easiest.

Grading

- The grade for the project proposal will be split approximately evenly between the design proposal and code sample.
- The design proposal will be graded on appropriate scope, degree of novelty, clarity of presentation, and how well it integrates a feature or features of the selected programming language.
- The code sample will be graded on the clarity and completeness of your video.
- Failure to follow the instructions in this document will result in proportionately-scoped grade deductions.

CLASS PRESENTATION

Each student will do a 5 to 5½ minute in-class presentation during the last week of class or the scheduled exam time, followed by a 1 to 1½ minute time for questions. Given the time constraints, this presentation will not be particularly in-depth, but you should include a very brief overview of your project for context (60-90 seconds), and then describe how you used a feature of the programming language you chose to complete it. Your presentation should clearly answer the question "*how is the programming language or framework you are using well or poorly suited to solve this problem?*" You may include visual aids such as a slide deck, code snippet, or demo, but are not required to do so (if you do use slides, a decent estimate is one slide per minute).

If working with a partner, double all times above, and divide your shared presentation slot roughly evenly between yourselves. Your team should highlight at least two features of the language or framework you're using during your presentation.

Given time constraints, this will not necessarily be a full "project presentation"; the goal of this presentation is highlight a programming problem motivated by your project, and demonstrate how that problem is solved in your chosen programming language. Focus your presentation time on the PL aspects of the project.

There will be up to 10 presentations per day on the following class dates (all times Pacific):

- Tuesday, November 30, 11:20am-12:45pm
- Thursday, December 2, 11:20am-12:45pm
- Monday, December 6, 10:30am-12:30pm

To encourage engagement with your classmates' work, there will be a forum thread on each presentation day with the following question: "*What is something you learned, thought was cool, or want to learn more about from the class presentations today?*" Any student who responds to one of these threads will get a 2% bonus on their own overall project grade, as will any students who ask questions on their classmates' presentations, to a maximum of 2% bonus per day.

Submission

- The primary evaluation for the class presentation is the presentation itself.
    - You may optionally pre-record your presentation and play it during your time-slot.
- You should also submit any slide decks or presentation notes in PDF format to the "Project Presentation" assignment dropbox on Moodle by the last day of presentations.
    - The format of this is up to you, but should include PDF export or screenshots of any visual aids you produce, and bullet-point or long-form notes for what you will say during your presentations.

Grading

The grade for the project presentation is based on the following factors:

- Content of presentation
    - Overview of project provides sufficient context to motivate presented feature.
    - PL feature(s) used are described.
    - Language domain-suitability question clearly answered.
- Quality of presentation
    - Presentation is clear, comprehensible, and flows well.

    - Presentation is practiced and smooth.
    - Any slides, demos, or other visual aids are informative and attractively-designed.
    - Presenter answers questions briefly but informatively.
- Follows assignment specification
    - Adheres to time limit (5 to 5½ minutes; going over-time will be more harshly graded than going under).
    - Notes submitted as described above.

FINAL SUBMISSION

The final project submission is due on the last day of classes (Friday, April 30) at 10pm Pacific. In addition to the code implementing your project, documentation is a heavily-weighted portion of the final project grade. Even relatively small projects like this require guidance for a developer unfamiliar with them (such as your course instructor) to read and understand them. As such, you will include a written report with your code submission which includes the following sections in the following order:

- **Overview**: A brief overview of the purpose of the project.
    - You may pull text from your project proposal for this if you wish.
    - Note any substantial deviations from the proposed project here.
- **Code highlights**: Point out some key elements of your project solution.
    - Copy at least three separate code snippets from your project, and write a paragraph for each describing:
        - What that code snippet means (its *semantics* in your language)
        - and how it implements a portion of your project.
    - Each code snippet should have its file and line noted so that the instructor can find it in your code submission.
    - Teams working together should copy at least six code snippets.
- **File listing**: This should include a brief description of the purpose of each file in the code submission.
    - The file descriptions should be in a bulleted tree.
    - It is acceptable to have a single-file submission.
    - Auto-generated or framework code (if present) can be described at the level of its containing directory.
    - The file that includes the entry point for the program (main function or its analogue) should be noted in this list, along with what construct in that file is the program entry point.

To demonstrate functionality of your code, you should also produce a short (2-5 minute) video demo of you compiling and running your code through at least one test case. Setting up a new toolchain takes a significant amount of time, so this allows you to provide a better demonstration of your program's functionality than the course instructor attempting to replicate your build.

Submission

- Your written report should be in PDF format, and include all the elements discussed above.
- The final project should be submitted to the "Project Submission" Moodle dropbox in the same format as the code sample in the project proposal.
  - In particular, multi-file submissions or those with significant directory structure should be archived in a .zip file.
  - If your submission exceeds the 10 MB file-size limit on Moodle, provide it to the instructor by an alternate mechanism (a Microsoft OneDrive shared folder is likely easiest).
- Either a link to the demo video or the video itself should be submitted on Moodle as well.
  - I believe students have access to the TechSmith screen-capture tool through UP at up.techsmithrelay.com, but you may use any tool you prefer to capture the demo.

Grading

- Roughly a third each of your project grade will depend on the written report, the code submission, and the demo video.
- The written report should be complete according to the specification above and both clear and concise (favour clarity over concision if you find the two in conflict).
- Your code will be examined, and should be clearly written, make appropriate use of whitespace, and use understandable identifiers. The code should be modularized in a language-appropriate fashion.
  - Be particularly generous with code comments, given that the instructor is not (likely) an experienced user of the language you chose.
  - Build artifacts (such as compiled executables) should not be included in your submission.
- The video should show you compiling your project code (if applicable to the language you chose) and running it through at least one test.
  - If your project has text-based inputs, either type them live or show the input file in the video.

Good luck! Contact the instructor if you have any questions.