CMPSC 220
Programming Languages Concepts
Fall 2015
Bob Roos
`http://cs.allegheny.edu/sites/rroos/cs220f2015`

Final Project
Proposals due Wednesday, 18 November
Progress Report due on Monday, 30 November
Project due on the last day of classes, Tuesday, 8 December

Overview

For your final project, you must explore a new programming language, write a number of programs in that language (including several of moderate difficulty and length), write a formal report about the language, and give an informal five-minute demonstration to the class. **You may work in teams of at most two.** If you work as a team you must also submit a "team report" that details the specific contributions of each team member. Each team should submit a single report (and separate team report) with both names given as authors and should do a joint presentation.

Your language must be one that you can demonstrate to me and to the class and one that I can use to test your programs. This means either that the language is installed on the Alden lab machines, or there is an online compiler or interpreter (e.g., "`ideone.org`" or a language-specific site such as "`jsfiddle.net`"), or there is an easily-installed compiler or interpreter that is freely available and does not require superuser permission or other special software to install and run. If I can't test your programs then I can't grade your project.

Details

While you are not expected to become fluent in all aspects of your chosen language, you are expected to demonstrate the features that would be commonly used by most programmers using the language as well as any features that might be unique to that language. This should be done by writing ***original*** programs and using them to describe or illustrate the language features.

You will certainly need to find sample programs on the Web or from other resources in order to teach yourself the language, and you may *significantly* adapt some of these (with appropriate acknowledgements of the original program authors and the source of the code) into your own examples, but under no circumstances should you simply include code that closely resembles a program written by someone else, even if the original source is cited. One minor exception: a "hello, world" type program may be adapted from another source (but still with appropriate acknowledgement).

Proposal (Due Weds., 18 Nov.)

Upload a document to your repository that includes:

- your name (and your partner's name if you are on a team)

- the language you wish to study
- details on language availability (e.g., "installed in Alden lab" or "available as a `.jar` file at `http://picard.net`" or "online compiler at `http://www.tutorialspoint.com/swift/`")
- why you chose this language

Each partner on the team should submit a separate proposal (but both names should appear in each one).

Progress Report (Due Mon., 30 Nov.)

Everyone must submit an individual progress report, even if part of a team. The progress report should be at least a page long and should include an informal description of what you have learned so far (e.g., you could informally answer some of the questions listed in the next section regarding scoping, etc.). It should also include at least one original, non-trivial code sample (in a separate file, ready to run) along with an explanation of what features are showcased in the example. This shows me that you are, in fact, writing programs in the language and learning about it. If you haven't yet successfully written a working program (beyond "hello, world") include your non-functioning code and a listing of the errors, together with a description of what you are attempting to do.

If you are part of a team, your progress report should describe your individual contribution to the project so far.

Report

The report is a formal document—it should have a title, numbered pages, section headings, numbered and labeled figures, a bibliography. One report is submitted for each team, with both team members listed as authors.

The report must be submitted as a PDF file—do not submit `.odt` or `.txt` files. (If you know LaTeX, you're in good shape! But you can usually do a "Save as" or "Export" to PDF in just about any editor.)

The content of your report should be guided by the topics we have studied in this course. For your chosen language:

- Give some brief background (who created it? when? why?).
- Explain its importance (where is it used? why is it worth studying?)
- Compare it to similar languages. In what ways is it similar to them? What distinguishes it from similar languages?
- How is it classified according to the criteria we've looked at, such as
  - compiled? interpreted?
  - paradigm (imperative? functional? object oriented? special purpose? ...)

- scoping (lexical? dynamic? block-level scoping? function-level scoping? ...)
- typing (statically typed? dynamically typed? ...)
- other criteria (for instance, parameter passing methods—by value? by reference? applicative/normal/lazy order? ...)

The bibliography should list the resources you used (books, online tutorials, other online resources such as stackoverflow.com, etc.) and these should be cited in the body of the paper.

Programs

Your programs should be in separate files, ready to compile and/or run. Each one should be fully commented and include your name, a line about Honor Code, a short description, citations to any relevant sources (e.g., was this program adapted from one you found elsewhere?), and other appropriate comments (e.g., highlighting language features that are being demonstrated).

You should also include snippets of programs in the written report whenever you need them to illustrate some feature of the language. Such snippets should be short, should be typeset in a fixed-width font (such as Courier), and should be set off in display style as numbered figures. For example,

> ...In the Picard language, the main program begins with the reserved word "`engage`" and is followed by one or more commands. The last line of every program is the keyword "`makeitso`." Figure 1 illustrates a typical Picard main program structure.

```
main() {
   engage;
   for i = 1 to math.infinity do {
      explore();
      seek();
      goboldly();
   }
   makeitso;
}
```

Figure 1: Typical Picard main program showing "`engage`" and "`makeitso`"