CSci 4651 Spring 2016

Final paper (50 points): study of programming languages design.

Due: Abstracts Tuesday April 26th, voting in on Th April 28th, and final papers due Thursday May 12th (the day of the final)

Several papers will be selected for presentation during the last week of classes based on voting. Others are welcome to present their papers at a scheduled time outside of the class. Presenting a paper results in extra credit (the exact amount is determined by the quality of the presentation). Being selected for presentation results in additional extra credit, whether you present or not (the amount depends on the number of votes).
Write a short (3-5 pages double-spaced, or equivalent) case-study of programming languages. You have three options to choose from:

- To compare two existing programming languages.
- To describe in detail an existing programming language. Generally it is expected that it is not one of the languages that we studied in class (Clojure, OCaml (ML), Ruby, C++, Java). If you are going in-depth into a particular feature, you can use one of the above languages, but you should talk to me first to make sure that your paper is sufficiently different from the material in the class.
- To write a brief proposal for a new programming language that you think would be an improvement over existing ones. You need to explain why this language is an improvement. Be specific about the intended usage of the language.

I will also consider other options as long as they address understanding the features below, just talk to me ahead of time if you have alternative ideas.

The languages that you consider (or design) must be Turing-complete. They may be general-purpose or special-purpose.

You need to address the following aspects of the language(s):

- the main programming paradigm or a combination of paradigms (imperative, functional, object-oriented, etc.)
- type discipline: is the language "more" typed or "less" typed (does it require type declarations, how freely can one combine types)?
- static vs. dynamic type system (or a combination?) What are guarantees about run-time type errors after static typechecking (if any)?
- support for type polymorphism: can you write a function that works for different types? Does the language support parametric types, such as Java generics or C++ templates? Is there overloading?
- is the language object-oriented? What is its inheritance model and how does it relate to subtyping?

- Is it statically or dynamically scoped?
- Is there a built-in support for concurrency?
- mention unusual features of the language, if any.
- what type of applications is this language used for? What are its strengths and weaknesses? Please justify your answer to this question by the information above.

The paper will be graded based on the quality of addressing the above questions, justification for your answers (you need to reference your sources or use your personal experience), and the clarity and quality of writing.