Homework 2: Functional Language Issues

Due: 1(a) on Feb 15.; rest of problem 1 on Feb. 20.

In this homework you will learn about design and/or research issues in functional programming.

You are to work in groups to solve this homework. Please follow the policy in section 6 of the course policies document.

For all Haskell programs, you must run your code with a Haskell interpreter (such as `hugs`). You must also provide evidence that your program is correct (for example, test cases). Hand in a printout of your code and your testing.

1. (200 points) You are to write a literate Haskell program that discusses and reacts to one of the following papers.

   - John Backus. Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs [Bac78].
   - Paul Hudak. Conception, Evolution, and Application of Functional Programming Languages [Hud89]. **Special:** Focus on one section of this paper for your "reaction" section; I suggest the section about I/O, to which you might want to compare the current version of Haskell.
   - Philip Wadler. The Essence of Functional Programming [Wad92].
   - Philip Wadler and Stephen Blott. How to make ad-hoc Polymorphism less ad hoc [WB89].
   - Mark P. Jones. A system of constructor classes: overloading and implicit higher-order polymorphism [Jon95].
   - Sheng Liang, Paul Hudak, Mark Jones. Monad Transformers and Modular Interpreters [LHJ95].
   - Martin C. Henson. Chapter 3 of Elements of Functional Languages [Hen87], which describes programming and verification. **Special:** Compare this with Chapter 9 of [Dav92].

   (Other papers may be substituted, with approval, but they must be published. That is, it is not acceptable to just get a paper off the web.)

   (a) (40 points) Pick your paper (see above), and have each group member read it. Each group member (individually!) should prepare a summary of the paper covering the following points. (Please clearly label each of these as a section in writeup.)

      - What is the problem(s) addressed by the paper? Do you think it is a significant problem? Why?
      - Describe the main points of the argument and the solution(s). This should be the longest section of your writeup. (Note the special remarks about some of the papers above, and take those into account.) Note any sloppiness in the paper's argument, places where you don't believe it, or where you believe it but not just because of what the author said.
      - How is this claimed to be an advance on related work? (For a survey paper, this point is moot unless certain ideas are claimed as new with that paper. If there are no such claims, then just say so.) Do you believe these claims? Why?

      For this part of the problem, save and hand in the individual summaries, unedited. These won't be graded, but are a check on your process.

   (b) Blend the individual summaries into a group summary, which you all agree upon. This is to be typed in the form of a literate Haskell program, of no more than 10 pages in length.

- (80 points) Be sure to address each of the points given above in a section clearly labeled as addressing that point.
  Your grade for this part will be based on: the clarity of your summary, its soundness (you start from facts or reasonable assumptions, your logic is convincing), its completeness (you consider relevant aspects of the problem and alternatives), and its conciseness and elegance (see [GS90] for a helpful tip).
  Be very careful to attribute any direct quotations and ideas, even from the paper you are summarizing. (Use quotation marks and citation forms [Van92].) Failure to do this may result is substantial loss of points.
- (20 points) Include translations into Haskell of the key examples that illustrate your summary. (For papers that already use Haskell, note what code is quoted.)
  The grade for this part will be based on: the correctness and clarity of the code. (If the author made something unclear say why, and show how to fix it.)
- (60 points) Also include, as a separate section in your writeup (in whatever order you want it), a section in which you write some literate Haskell code which either applies or extends the author's ideas. For example, if the author has proposed a technique, and illustrated it with an example, you could make up a new example, or extend the author's example (making it bigger, more realistic, etc.). If the author has proposed some language feature, and that feature is implemented in Haskell, then you could use the feature to write some example.
  The grade for this part will be based largely on the creativity and interest of your example. It will also take into account any mistakes or unclear things in your code. Be sure to use the literate style, which puts explanation around the code. (A good choice is to explain first, then show the code.)

(c) (10 points) Hand in the above, with the group leader's certification as described in section 6.3 of the course policies document. (Your individual answers for part (a) above serve the role of the "individual plans" called for there.)

2. (32 points; extra credit) (You may do the extra credit problems either by yourself, or in your group.) Based on your reading, how practical do you think functional programming is for:
   - everyday use by programmers?
   - use in numerical computation (such as weather forecasting, large physics simulations, etc.)?
   - use in database systems as a query language?
   - work in language semantics?

   Describe and illustrate (with Haskell code if possible) the reasons for your answers.

3. (20 points; extra credit) If you have done OO programming. What do you think the potential for reuse is of functional programming vs. OO programming? Describe and illustrate (with Haskell code if possible) the reasons for your answers.

References

[Bac78] John Backus. Can programming be liberated from the von neumann style? a functional style and its algebra of programs. *Communications of the ACM*, 21(8):613–641, August 1978.

[Dav92] Anthony J. T. Davie. *An Introduction to Functional Programming Systems Using Haskell.* Cambridge Computer Science Texts. Cambridge University Press, New York, N.Y., 1992.

[GS90] George D. Gopen and Judith A. Swan. The science of scientific writing. *American Scientist*, 78:550–558, November-December 1990.

[Hen87] Martin C. Henson. *Elements of Functional Languages.* Blackwell Scientific Publications, Oxford, England, 1987.

[Hud89] Paul Hudak. Conception, evolution, and application of functional programming languages. *ACM Computing Surveys*, 21(3):359–411, September 1989.

[Jon95] Mark P. Jones. A system of constructor classes: overloading and implicit higher-order polymorphism. *Journal of Functional Programming*, 5(1):1–35, Jan 1995. An earlier version appeared in FPCA '93.

[LHJ95] Sheng Liang, Paul Hudak, and Mark Jones. Monad transformers and modular interpreters. In *Conference Record of POPL '94: 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Francisco, California*, pages 333–343, January 1995.

[Van92] Mary-Claire Van Leunen. *A handbook for scholars.* Oxford University Press, New York, N.Y., revised edition, 1992.

[Wad92] Philip Wadler. The essence of functional programming. In *Conference Record of the Nineteenth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 1–14. ACM, January 1992.

[WB89] Philip Wadler and Stephen Blott. How to make ad-hoc polymorphism less ad hoc. In *Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas*, pages 60–76. ACM, January 1989.