

Advanced Programming Languages

Homework Assignment 1F and 1C

EECS 590

Logistics. You must work alone. Your name and Michigan email address must appear on the first page of your PDF submission but *may not appear anywhere else*. This is to protect your identity during peer review. The first page of your submission is *not* shared during peer view but all subsequent pages are.

Exercise 1F-1. Bookkeeping [2 points]. These answers should appear on the first page of your submission and are kept private.

1. Indicate in a sentence or two how much time you spent on this homework.
2. Indicate in a sentence or two how difficult you found it subjectively.

All subsequent answers should appear after the first page of your submission and may be shared publicly during peer review.

Exercise 1F-2. Language Design [5 points]. Comment on some aspect from Hoare's *Hints On Programming Language Design* that relates to your programming experience. Provide additional evidence in favor of one his points and against one of his points. Do not exceed three paragraphs. Both your ideas and also the clarity with which they are expressed (i.e., your English prose) matter. Readers should be able to identify your main claim, the arguments you are bringing to bear, and your conclusion.

Exercise 1F-3. Simple Operational Semantics [3 points]. Consider the IMP language discussed in class, with the Aexp sub-language extended with a division operator. Explain what changes must be made to the operational semantics (big-step only). Write out formally any new rules of inference you introduce.

Exercise 1F-4. Language Feature Design, Large Step [10 points]. Consider the IMP language with a new command construct “`let $x = e$ in c` ”. The informal semantics of this construct is that the Aexp e is evaluated and then a new local variable x is created with lexical scope c and initialized with the result of evaluating e . Then the command c is

evaluated. We also extend IMP with a new command “`print e`” which evaluates the Aexp e and “displays the result” in some un-modeled manner but is otherwise similar to `skip`.

We expect (the curly braces are syntactic sugar):

```
x := 1 ;
y := 2 ;
{ let x = 3 in
  print x ;
  print y ;
  x := 4 ;
  y := 5
} ;
print x ;
print y
```

to display “3 2 1 5”.

Extend the natural-style operational semantics judgment $\langle c, \sigma \rangle \Downarrow \sigma'$ with one new rule for dealing with the `let` command. Pay careful attention to the scope of the newly declared variable and to changes to other variables.

Exercise 1F-5. Language Feature Design, Small Step [10 points]. Extend the set of redexes, contexts and reduction rules for the contextual-style operational semantics that we discussed in class to account for the `let` command introduced above.

Exercise 1C. Language Feature Design, Coding. Download the Homework 1 code pack from the course web page. Modify `hw1.ml` so that it implements a complete interpreter for IMP (including `let` and `print`). Base your interpreter on IMP’s large-step operational semantics. The `Makefile` includes a “`make test`” target that you should use (at least) to test your work.

Modify the file `example-imp-command` so that it contains a “tricky” terminating IMP command that can be parsed by our IMP test harness (e.g., “`imp < example-imp-command`” should not yield a parse error).

Submission. Turn in the formal component of the assignment (1F-1 through 1F-5) as a single PDF document via the `gradescope` website. Your name and Michigan email address must appear on the first page of your PDF submission but may not appear anywhere else. Turn in the coding component of the assignment (1C) via the `autograder.io` website.