Homework 9

Due Thursday, 4/11/2019

Several questions on this homework involve ML programs. However, they behave identically to equivalent Haskell programs. If you have any problem understanding them, please let me know.

You will be turning in two files for this assignment, named `Hmwk9.pdf` and `Hmwk9.hs` They should be turned in as usual at `https://submit.cs.pomona.edu/2019sp/cs131`

The Haskell file should include only the program called for in problem 3b. If that file contains any test code, please comment it out before turning it in as there may otherwise be name clashes with my test code.

The grading for this assignment will be based on both the `Hmwk9.pdf` and `Hmwk9.hs` files that you turn in

1. The `Hmwk9.pdf` file should contain the complete solutions to all problems (including the Haskell program called for in problem 3b).
2. The `Hmwk9.hs` file should contain the program for problem 3b. The first line should be:

   ```
   module Hmwk9 where
   ```

If the Hmwk9.hs file runs without crashing on the test file given in problem 3b, *and* you correctly spell your name on both the hs and pdf files, you will receive 10 points credit (whether or not the answer printed is correct). All of the other points listed (including the first 20 points listed for problem 3) will be based on the correctness of the materials in the pdf file.

In particular, if you spell your name correctly in both files, and the program for 3b runs without crashing, but you neglect to include the code in the pdf file, then you will receive 10 points for running, but lose all 10 points for the correctness of your code. Please be careful to follow the instructions.

1. (15 points) **Exceptions and Recursion**

   Please do problem 8.4 from Mitchell, page 230.

2. (10 points) **Tail Recursion and Exception Handling**

   Please do problem 8.5 from Mitchell, page 230.

3. (20+10 points) **Modularity of Concrete Data Types**

   In lectures 12 and 13 we discussed an environment-based interpreter for PCF. It is available online from the course web page by following the link to "Programs from lecture". In this problem, I would like you to modify that program in order to determine how easy it is to add new features.

   (a) Please add a new function `prettyprint` that takes a PCF term and returns a string containing a nice readable version of the term. It should have type `prettyprint :: Term -> [Char]`

   Note: The data type `Term` is given in the PCF parser, available from the same programs page. For this problem we are going to ignore the PCF parser. Thus remove the import of ParsePCF from the interpreter code. To make up for this you will need to copy in the definition of Term:

```
data Term = AST_ID String | AST_NUM Int | AST_BOOL Bool
          | AST_FUN (String,Term) | AST_APP (Term, Term)
          | AST_SUCC | AST_PRED | AST_ISZERO
          | AST_IF (Term, Term, Term) | AST_REC (String, Term)
          | AST_ERROR String deriving (Show,Eq)
```

Note that when you do part b of this problem you will need to add ***AST_SUM*** *(see part b) to the export list of that module.*

The string returned by prettyprint should contain parentheses to indicate precedence. For example, the term

```
AST_IF(AST_BOOL True,AST_NUM 5, AST_APP(AST_SUCC, AST_NUM 2))
```

should be pretty-printed as

```
if True then 5 else (succ 2)
```

whereas

```
AST_APP(AST_IF(AST_BOOL True,AST_PRED, AST_SUCC), AST_NUM 2)
```

should be pretty-printed as

```
(if True then pred else succ) 2
```

(b) Please add a new term to PCF representing sums of integers. I.e., add a new clause to the definition of term of the form `AST_SUM (Term, Term)` so `AST_SUM(a,b)` represents `a + b`.

Modify the previous program that contains both the `newinterp` function to evaluate terms and `prettyprint` to accommodate the new term.

Please do not turn in the results of part a. I only want a single program that has the functionality specified in parts a and b. That is, the function `newinterp :: Term -> Env -> Value` should handle `AST_SUM` terms and the prettyprint function should be included in the file (and pretty-print should handle `AST_SUM`).

When done, please test your program with a file with the following contents:

```
import Hmwk9

 fc = AST_APP(AST_SUCC, AST_SUM(AST_NUM 5, AST_NUM 3))

 main :: IO()
 main = do
     putStr ("The value of " ++ (prettyprint fc))
     putStrLn (" is " ++ show (newinterp fc []))
```

If it works properly, it should print out: `The value of succ (5 + 3) is NUM 9`. Obviously you should test your program on more than this, but checking this will at least ensure that your program will not crash my test code for your program (keeping you from getting your 10 points).

(c) I am not asking you to modify the PCF parser code that is available on-line. However, please describe at a high level what would have to be done to modify that code.

(d) Discuss briefly the different impacts of adding a new function (like `prettyprint`) versus adding a new term (like `AST_SUM` to the data type in the previous two parts of this problem.

Note: If you wish, you need only turn in the code from the part b of this question rather than separate code for each of parts a and b. Of course you should also include your answers for parts c and d.

4. (25 points) **"Growing a Language" Reading**

Please read the short paper entitled "Growing a Language" from the auxiliary reading page. The author, Guy Steele, is one of the designers of Java, and this was his keynote talk at the major American conference on object-oriented languages (OOPSLA) in 1998. You may find it even more interesting to watch his talk on-line (see the link on the auxiliary reading page). See if you can figure out what he is doing before he explains it!

This paper includes a discussion of why some items were originally left out of Java when they designed the language, but likely should have been included. It also gives some insight into programming language design in general. The following questions will guide you through the paper:

(a) What is wrong with designing/using a small language like Lisp?

(b) What is wrong with designing/using a huge language (C++)?

(c) What was Steele's goal in designing Java?

(d) Why does Steele feel that overloaded operators are important. Contrast this with his conviction (expressed elsewhere) that overloaded methods might have been left out without much harm.

(e) Discuss why generics might have been left out originally, but why he now feels they are important.

Write a separate paragraph discussing each of these items. A few sentences on each part is sufficient.

We will spend some time discussing these points after the homeworks are turned in.

5. (15 points) **Smalltalk Reading**

Read "Design Principles Behind Smalltalk" by Dan Ingalls (available on the "links" page).

Think about the Principles set forth. Which are new? Which have we seen before? What principles are remniscent of Lisp? What is the scope of the language (ie, what is included in its definition)? A few sentences on each item is sufficient— clarity is more important than length of answer.