SOFTWARE ENGINEERING PROGRAMME
UNIVERSITY OF OXFORD
www.softeng.ox.ac.uk

ASSESSMENT

Student: Mayur Pant

Course: Functional Programming

Date: 15th September 2014

Grade: 78

Jeremy Gibbons; Jose Pedro Magalhaes December 11, 2014

REPORT

This was very well done. You've clearly grasped basic functional programming, with recursive functions over structured data. You also coped well with the higher-order functions needed for shallow embeddings. You didn't manage Q10, the trickiest; I've given you the first half, and maybe you can now complete the second.

Your submission was very nicely structured - you've taken care to think about the convenience of the reader, grouping together similar questions (on the other hand, I have to say that reordering the questions makes it a little harder for me to mark...).

You've included a section on what you learned through the exercise; I'm glad you liked the XML question. I'm sorry you couldn't find much about deep and shallow embeddings apart from my own work; in fact, this is a current research topic, and hot off the press. I hope your colleague Emma liked her Latin cards.

Question-by-question comments follow.

QUESTION 1

This was perfect. I'm not sure how you would make your "otherwise" clause work: you can't match on several patterns simultaneously and also extract the |x| field in both cases.

(Incidentally, one says that a type such as |Dir| is a *member* or *instance* of a typeclass such as |Show|.)

QUESTION 2

Fine. Well done for spotting that you can make |Dir| an instance of |Enum|. Your function would be a bit neater if you had defined an auxilliary function to turn the direction by 90 degrees, avoiding two |Turn| clauses.

QUESTION 3

Perfect. I'm not sure your alternative definition is an improvement; it may save some keystrokes, but you want to take into account both brevity and simplicity, and I'd say it's a bit more complicated.

QUESTION 4

The first answer is perfect. But I think the |eval| and |lit| bits are irrelevant.

QUESTION 5

Perfect - and your |succ_wrap| function is nice, making good use of type classes. (Can you see how to do it without requiring |Eq|?)

QUESTION 6

You started by trying to reverse-engineer Q10; that would work in principle, but I don't think you quite got to the bottom of it (nothing here specifies "pos semantics"). However, your second "researched" attempt was perfect.

QUESTION 7

You missed my point here. The shallow embedding |Dir2| isn't sufficient, because it doesn't support `turning' as we need for functions |dir| and |pos|: if directions are represented as functions, we can't directly inspect them to change from one function to another. But there is a clever way to indirectly inspect |Dir2| functions, by rotating the coordinate space - do you see how?

QUESTION 8

This was good; you presented a number of alternatives, and spotted that it is worth defining a local variable to avoid recomputing |forwards' d p|. Note that computing |b == True| is redundant, just as |x+0| is.

QUESTION 9

As with Q6, your first attempt is incomplete, not providing any of the logic for plotting. But your second "researched" method is perfect.

QUESTION 10

You make a little headway with this question, but not much. Here's how to extract the "pos" semantics from a program like the one in the question:

```
> posS y = y ( \ x -> \ p d -> x (forwards d p) d,
> \ x -> \ p d -> x p (turnLeft d),
> \ x -> \ p d -> x p d,
> \ p d -> p)
```

Having seen this, can you now extract the "plot" semantics from *the same program*?

QUESTION 11

This was good. Well done on finding a paper of mine online with part of a solution! It's good that you cited this properly, and also explained how it works. Your |picture_lines| function would have been a bit neater using |map| rather than |fold|; and it would be better to compute the picture dimensions from the picture provided, rather than hard-wiring them.