Programming Paradigms 2023
Session 2: First steps in Haskell

Problems for solving and discussing

Hans Hüttel

12 September 2023

1 Problems that everyone should solve during the session

For this session and the sessions that follow we will use the ghci interpreter for loading and running Haskell code.

1. ***(Everyone at the table)*** Use the functions in Section 2.4 to define a function allbutsecond that will, when given a list list , return the list consists of all but the second element of the list. As examples of what this function should do, we expect that

   ```
   allbutsecond [1,4,5,6]
   ```

   should give us [1,5,6] and that

   ```
   allbutsecond ["some, " bizarre", "mango'']
   ```

   will give us ["some","mango"]. How can you become more certain that your solution is correct?

2. ***(Pair programming*** Use the functions in Section 2.3 to define a function midtover that will, when given a list list of length n, return a pair of two lists (list1 , list2) such that list1 consists of the first $\lfloor \frac{n}{2} \rfloor$ elements from list1 and such that list2 consists of the remaining elements. *Hint:* Please use integer division – this is the ‘div‘ function – please remember the backquotes if you want to use it as an infix operator; the prefix version is called (div).

   As examples of what this function should do, we expect that

   ```
   midtover [1,4,5,6]
   ```

   should give us [(1,4],[5,6]) and that

   ```
   midtover ["this, "is", "actually'', "a'',''fairly'',''long'',''
       list'']
   ␣␣␣␣
   ```

   will give us (["this,"is","actually "],["a ","fairly ","long ","list "])

   How can you become more certain that your solution is correct?

3. ***(Pair programming*** Something is wrong in the following tiny piece of code. But what is wrong? Explain. Then repair the code such that it works.

   ```
   bingo (x,y) = x mod z
   where
   z = y + 42
   ```

More problems to solve at your own pace

a. There is a function called reverse in the Haskell prelude that allows us to reverse any list. Use reverse to give a definition of a function final that returns the last element of a given list.

b. How can we change qsort from simple.hs such that it sorts a list in *descending* order? There are more ways of approaching this. Do the calls

```
qsort [3,4,57]
```

and

```
qsort ["plip","plop","abba"]
```

both make sense? Why/why not?

c. Suppose we changed the definition of qsort from the file simple.hs from Session 1 such that we replaced <= to <. What would happen then?