List Induction: Tutorial sheet 2

Jeremy Bradley

16 January 2005

Assessed Exercise 2: Question **3** is assessed and is due in to the SAO by 4.30pm on **1 February 2005**. This is a hardcopy submission but you still need to register your submission using CATE which will also provide you with your submission cover sheet: `https://sparrow.doc.ic.ac.uk/~cate/`

1. The post-condition in the function `sumExample` is incorrect for one value of `x`. Suggest a modification to the post-condition so that it is correct for all `x`-values and prove the new post-condition by induction.

```
-- Pre-condition: n >= 1
-- Post-condition: sumExample n x = (x^(n+1) - x) / (x-1)
sumExample :: Int -> Int -> Int
sumExample 1 x = x
sumExample n x = x + (x * (sumExample (n-1) x))
```

2. (a) The power set of a set, S, is the set of all subsets of S. For example, the power set of $\{1, 2\}$ is $\{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$, where $\emptyset$ is the empty set. Complete the following function which takes an input set and calculates its power set.
[Here we use a Haskell list to represent a set and the order in which the set is generated is not important.]

```
powerSet :: [a] -> [[a]]
powerSet [] = [[]]
powerSet (x:xs) = ...
```

(b) Show by induction on the input list of `powerSet` that

$$\texttt{length}\ (\texttt{powerSet xs}) = 2^{(\texttt{length xs})}$$

You may use the length definition and property from question 3 without proof.

3. **ASSESSED** Given the following definition of the `length` function,

```
length :: [a] -> Int
length [] = 0
length (x:xs) = 1 + (length xs)
```

Prove that for all lists `xs`, `ys`:

$$\texttt{length}\ (\texttt{xs++ys}) = (\texttt{length xs}) + (\texttt{length ys})$$

[Hint: you only need to perform an induction on one list variable, say `xs`]

4. Prove that `foldrMin` produces the minimum element of a list of integers.

```
foldrMin :: [Int] -> Int
foldrMin [] = error "no elements in input"
foldrMin (x:[]) = x
foldrMin (x:xs) = min x (foldrMin xs)
```

5. [This is slightly different from previous inductions you may have seen, as it involves induction with a picture. The induction structure is still the same though.]

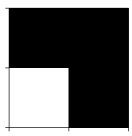

Figure 1: 1 L-shaped piece in a 2×2 grid

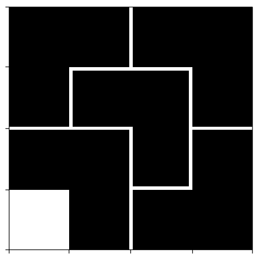

Figure 2: 5 L-shaped pieces in a 4×4 grid

(a) Given an L-shaped tile, shown if Fig. 1, show using induction on n that a $2^n \times 2^n$ grid can be filled with L-shaped tiles in such a way that a square is left empty in the bottom left hand corner of the grid. The cases $n = 1$ and $n = 2$ are shown in Fig. 1 and Fig. 2, respectively.

(b) State a formula in terms of n for the number of L-tiles in a $2^n \times 2^n$ grid.

(c) Using your understanding of the induction step from part (a), complete the following Haskell function for recursively calculating the number of L-tiles in a $2^n \times 2^n$ grid. Prove by induction that your answer to part (b) is a post-condition for the function, `numTiles`.

```
numTiles :: Int -> Int
numTiles 1 = 1
numTiles n = ...
```