Mathematical Induction: Tutorial sheet 1

Jeremy Bradley

10 January 2005

Assessed Exercise 1: Question **3** is assessed and is due in to the SAO by 4.30pm on **25 January 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. Prove using induction that $\sum_{i=0}^{n} i^2 = \frac{n}{6}(n+1)(2n+1)$ for $n \geq 0$.

2. (a) Prove that for all $n \geq 0$, (`proc1 n`) ‘`mod`‘ `6` $= 0$ is a property of the function `proc1`.

```
proc1 :: Int -> Int
proc1 n = n^3 - 25 * n
```

   (b) Using the result from part (2a), (without using induction) show that, if the restriction $n \geq 0$ is removed, (`proc1 n`) ‘`mod`‘ `6` $= 0$ is true for all integers n.

   (c) Prove that for all $n \geq 0$, all elements of the list generated by `procInts n` are divisible by 6 (again using the result from part (2a)).

```
-- pre-condition: n >= 0
procInts :: Int -> [Int]
procInts 0 = [0]
procInts n = (proc1 n : procInts (n-1))
```

3. **ASSESSED** Given the following function definition, prove that for all $n \geq 1$:

$$\texttt{head (squareList n)} = n^2$$

```
squareList :: Int -> [Int]
squareList 1 = [1]
squareList n = (2*n + m - 1 : ms) where
    ms = squareList (n-1)
    m = head ms
```

4. Consider the program:

```
-- pre-condition: n >= 1
uList2 :: Int -> Int
uList2 1 = 3
uList2 2 = 5
uList2 n = (3*uList2 (n-1)) - (2*uList2 (n-2))
```

   (a) What is the post-condition for `uList2 n` in terms of n?

   (b) Prove that your post-condition holds by induction.

5. Given the following program for calculating powers of 2:

```
-- pre-condition: n >= 0
power2 :: Int -> Int
power2 0 = 1
power2 n = 2 * (power2 (n - 1))
```

   (a) Prove the property that `power2 n` $< n!$ for all $n \geq 4$.

   (b) Given the following more efficient implementation of `power2`, prove the same property of `power2mod n` for $n \geq 4$.

```
-- pre-condition: n >= 0
power2mod :: Int -> Int
power2mod 0 = 1
power2mod n
    | (mod n 2) == 0 = (power2mod (div n 2))
                       * (power2mod (div n 2))
    | otherwise      = 2 * power2mod (n - 1)
```