Starred exercises are optional.

1) • Give the intermediate configurations showing that indeed

$$(s, \vdash 0010\sqcup^{\infty}, 0) \xrightarrow[M]{*} (t, \vdash 0011\sqcup^{\infty}, 3)$$

as claimed on slide 11 of the lecture of week 3. How many steps were needed?

• Represent the TM on slide 9 of the lecture of week 3, as a TM in the sense of the Turing machine simulator demonstrated in the lecture (`http://morphett.info/turing/turing.html`). Using it, simulate the computation in the first item.

2) Represent the *square* function $\{(0,0),(1,1),(2,4),(3,9),(4,16),\ldots\}$ on natural numbers:

• either as a Turing Machine with as input alphabet $\Sigma = \{1\}$, that is, numbers are represented in unary, e.g. an input 9 is represented by putting 111111111 on the tape. Use the Turing machine simulator to show that your TM is correct, in particular that on input 5, i.e. 11111, the TM halts with 25 on the tape, i.e. with the tape containing the left-end marker followed by a string of 25 1s and then a blank.

• or as a functional program (either a Haskell program or a finite number of equations as on slide 14 of the lecture of week 3) that instead of using some 'built-in' natural numbers, uses a unary representation as 'successors of zero'. That is, in Haskell, we could define:

```
data Nat = Zer | Suc Nat deriving Show
```

Functions can then be defined by distinguishing cases on whether the input is 0 (`Zer`) or of shape $1+n$ (`Suc n`). For instance, the doubling function may be defined by:

```
double Zer = Zer
double (Suc n) = Suc (Suc (double n))
```

Indeed `double (Suc (Suc Zero))` yields `Suc (Suc (Suc (Suc Zero)))`.

or (*) both.

3) Say a list $l = [n_1, \ldots, n_k]$ of natural numbers $n_1, \ldots, n_k \in \mathbb{N}$ is *locally* $\leq$-sorted if $n_i \leq n_{i+1}$, and *globally* $\leq$-sorted if $n_i \leq n_j$, for $i, j \in \{1, \ldots, k\}$ with i less than j (Updated 18-10-2019).

• Argue that local $\leq$-sortedness implies global $\leq$-sortedness;

• Give a counterexample showing that local R-sortedness need not imply global R-sortedness if we would replace the natural order $\leq$ in the above by an arbitrary relation R. (Give both a suitable relation R and a list l that is locally R-sorted but not globally R-sorted.)

(∗) Argue (informally) that for insertion sort (see also this visualisation) to produce a locally R-sorted list, it suffices that R is *total* in the sense that for all x, y, $x\ R\ y$ or $y\ R\ x$ (so it is not necessary that R be a partial order; in particular R need not be transitive).

4∗) Write a total Turing Machine that accepts numbers written using *quits* (the digits until four: 0, 1, 2, 3; that is, these are base-4 numbers) that are divisible by 3, and rejects otherwise. For instance, the number 322 corresponds to $3\cdot4^2+2\cdot4^1+2\cdot4^0 = 58$ which is not divisible by 3 so should be rejected, but 222 is divisible by 3 as it corresponds to $2\cdot4^2+2\cdot4^1+2\cdot4^0 = 42 = 14\cdot3$ so should be accepted. What is the complexity of your algorithm, expressed in the number n of quits of the input?