Assignment 3

Deadline for solutions: 9.12.2019

Exercise 1 Braun Trees and Merge Sort (9 Points)

1. Formalize the ordering property of Braun trees and prove that bt-insert preserves it.

Hint: One possible way to formulate the requested ordering property is in two stages as follows:

```
-- the top element is greater than the given element
-- and for every subtree the top element is greater than the descendants
bt-ord-h : ∀ {n : ℕ} (a : A) (t : braun-tree n) → Set
bt-ord-h _ bt-empty = ⊤
bt-ord-h a (bt-node b l r _) = a ≤A b ≡ tt ∧ bt-ord-h b l ∧ bt-ord-h b r


-- for every subtree the top element is greater than the descendants
bt-ord : ∀ {n : ℕ} (t : braun-tree n) → Set
bt-ord bt-empty = ⊤
bt-ord (bt-node a l r _) = bt-ord-h a l ∧ bt-ord-h a r
```

2. `list-merge-sort.agda` of the Iowa Agda library contains an implementation of merge sort using Braun trees. Prove that the length of the input list is equal to the length of the output list.

3. Analogously, formalize and prove that the resulting list, sorted by merge-sort is indeed sorted.

Hint: You need to assume totality of the comparison relation $\leq$, i.e. that the type total $_{\leq}A$ is inhabited. This can be done elegantly using the module mechanism (as e.g. in `bst.agda`). Consider proving the following lemma:

```
sorted : ∀ (l : 𝕃 A) → Set
sorted l = ∀ (i : ℕ) (a b : A)
  → just a ≡ nth i l
  → just b ≡ nth (suc i) l
  → (a ≤A b) ≡ tt

merge-sorted : ∀ (l r : 𝕃 A) → (sorted l) → (sorted r) → sorted (merge l r)
```

(most of the complexity is concentrated in the base of induction, which amounts to an extensive case distinction over possible mutual arrangements of the initial list elements.

Exercise 2 Automation through Reflection (15 Points)

The point of the present exercise is to build a simplifier for integer expressions, analogously to the simplifier for list expressions from the lecture.

1. Use the following type for representing integer expressions build of zero, one, negation, summation and multiplication correspondingly over the expressions of the form $z\langle n\rangle$ where z is an element of $\mathbb{Z}$ and n is a *priority* parameter for rearranging sums and products by commutativity.

```
data ℤʳ : Set where
  _⟨_⟩ : ℤ → ℕ → ℤʳ
  0ʳ  : ℤʳ
  1ʳ  : ℤʳ
  -ʳ_ : ℤʳ → ℤʳ
  _+ʳ_ : ℤʳ → ℤʳ → ℤʳ
  _×ʳ_ : ℤʳ → ℤʳ → ℤʳ
```

2. Define the *semantic map*

```
ℤ⟦_⟧ : ℤʳ → ℤ
```

sending expressions to the corresponding values in $\mathbb{Z}$ in the expected way (and removing the priorities).

3. Implement a one-step simplifier `ℤ-simp-step` analogous to `𝕃ʳ-simp-step`, performing the following arithmetic simplifications:

$$x + 0 \to x \qquad x + 0 \to x \qquad (x + y) + z \to x + (y + z)$$

$$x \times 1 \to x \qquad x \times 1 \to x \qquad (x \times y) \times z \to x \times (y \times z)$$

$$x \times (y + z) \to x \times y + x \times z \qquad (x + y) \times z \to x \times z + y \times z$$

$$-(x + y) \to (-x) + (-y) \qquad (-x) \times y \to -(x \times y) \qquad x \times (-y) \to -(x \times y)$$

$$-0 \to 0 \qquad -(-x) \to x \qquad z\langle n\rangle + (-z\langle m\rangle) \to 0 \qquad (-z\langle n\rangle) + z\langle m\rangle \to 0$$

$$z\langle n\rangle + ((-z\langle m\rangle) + y) \to y \qquad (-z\langle n\rangle) + (z\langle m\rangle + y) \to y$$

and moreover swapping the arguments in $z_1\langle n\rangle + z_2\langle m\rangle$ and $z_1\langle n\rangle \times z_2\langle m\rangle$ if the priority m is strictly greater than n. Add further rules to ensure that the latter rearrangements remain stable under associativity of $+$ and $\times$.

4. Define a simplification function

```
ℤ-simp : (t : ℤʳ) → ℕ → ℤʳ
```

analogous to `𝕃ʳ-simp` from the lecture. Postulate the soundness property `ℤ-simp-sound : (t : ℤʳ) (n : ℕ) → ℤ⟦ t ⟧ ≡ ℤ⟦ ℤ-simp t n ⟧`without a proof.

5. Prove the property of 2×2 matrices $\det(M * N) = (\det M) * (\det N)$ from the previous assignment by normlizing the left and the right hand side using `ℤ-simp`. **Attention:** normalization does not yield the same value, for the left and the right hand sides – the resulting expressions still must be further rearranged to obtain an identity.

Exercise 3 Formal Type Theory (6 Points)

Using the formal rules of Martin-Löf type theory, show the following.

1. A definitional uniqueness principle for the unit type is not stated, however, the propositional version of it, stating that every element of the unit type is (propositionally) equal to $\star$ is provable. Formalize and prove it.

2. Derive proofs of symmetry and transitivity of propositional equality from the primitive rules from the lecture.