Assignment 4

Deadline for solutions: 7.01.2019

Exercise 1 Without K (16 Points)

Implement solutions to the following problems in Agda with the pragma `{-# OPTIONS --without-K #-}` activated. This corresponds to the general version of Martin-Löf type theory with the elimination principle for the identity types, as explained at the lecture. As a result, proofs of identities themselves become subject to nontrivial proofs. The following intuition is helpful when working with such proofs. You can think of $p : x \equiv y$ as a *path* from x to y on a surface. Then $\mathit{refl} : x \equiv x$ is a one point path, symmetry produces a reversed path $(\mathit{sym}\ p) : y \equiv x$, and transitivity concatenates two paths. For example, you can show that $\mathit{trans}\ p\ (\mathit{sym}\ p) \equiv \mathit{refl}$ (do it!). This is called the *groupoid interpretation* of type theory. The following variant of the identity type eliminator

```
J' : ∀ {A : Set ℓ} {x : A} (P : (z : A) → x ≡ z → Set ℓ) →
    P x refl → (y : A) (x≡y : x ≡ y) → P y x≡y
J' P p ._ refl = p
```

can thus be regarded as *(based) path induction*: to show a property $P\ y\ x{\equiv}y$ of a path $x{\equiv}y$, we show $P\ x\ \mathit{refl}$ (induction base) and that all paths $P\ z\ x{\equiv}z$ can be formed (so, we can continuously move from $z := x$ to $z := y$).

A type is *contractible* if it provably has exactly one inhabitant; a type is a *proposition* if all its inhabitants are equal; a type is a *set* if there is at most one proof of equality of any two its inhabitants. This is formalized in Agda as follows:

```
isContr : Set ℓ → Set ℓ
isContr A = Σ A (λ x → ∀ y → x ≡ y)

isProp : Set ℓ → Set ℓ
isProp A = (x y : A) → x ≡ y

isSet : Set ℓ → Set ℓ
isSet A = (x y : A) → isProp (x ≡ y)
```

1. Show that every contractible type is a proposition and every proposition is a set.

Hint: Second property is non-tivial and requires some exploration of the space of identity proofs $p : x \equiv x$. The idea is to prove that every proof $x{\equiv}y : x \equiv y$ is equal to the canonical proof witnessing isProp A. As an intermediate step, show the following, using (based) path induction:

```
prop-refl-prop : ∀ {A : Set ℓ}{x : A} (p : isProp A) → ((trans (p x x) (sym (p x x))) ≡ (p x x))
```

2. Show that a type A is a proposition iff every type $x \equiv y$ with $x\ y$: A is contractible.

3. Show that a type A is a set iff it satisfies the *K rule*, iff it satisfies *uniqueness of identity proofs*:

$$\mathsf{K} : \forall\, (A : \mathsf{Set}\ \ell)\ (x :\ A)\ (P : x \equiv x \to \mathsf{Set}) \to P\ \mathsf{refl} \to (x{\equiv}x :\ x \equiv x) \to P\ x{\equiv}x$$

$$\mathsf{UIP} : \forall\, (A : \mathsf{Set}\ \ell) \to \mathsf{Set}\ \ell$$

Hence, removal of the `{-# OPTIONS --without-K #-}` is precisely equivalent to stating that every type is a set. This explains the historical choice of the name Set for types in Agda.

4. Show that $\mathbb{B}$ and $\mathbb{N}$ are sets.

Hint: The second property is non-trivial and can be proven by induction over natural numbers, for which you will need to prove the following auxiliary property by path induction

$$\mathsf{pre} : \mathbb{N} \to \mathbb{N}$$
$$\mathsf{pre}\ \mathsf{zero} = \mathsf{zero}$$
$$\mathsf{pre}\ (\mathsf{suc}\ n) = n$$

$$\mathsf{h} : \forall\, \{x\ y : \mathbb{N}\}\ (x{\equiv}y : \mathsf{suc}\ x \equiv \mathsf{suc}\ y) \to \mathsf{cong}\ x{\equiv}y\ (\lambda\ z \to \mathsf{suc}\ (\mathsf{pre}\ z)) \equiv x{\equiv}y$$

(you will need to adapt `cong` from `eq.agda` and possibly other functions about equalities.)

Exercise 2 GCD (7 Points)

Greatest common divisor $gcd(a, b)$ of two natural positive (!) numbers is inductively defined as $gcd(a - b, b)$ if $a > b$, as $gcd(a, b - a)$ if $b > a$ and as a if $a = b$.

1. Implement gcd in Agda using the modules of Iowa Agda library. To that end you will need to design a corresponding termination proof.

Hint: A concise and elegant solution can be obtained by using the *lexicographic order* and a corresponding termination proof `↓-lex` from `termination.agda`. Note, however, that it is only one possible approach.

2. Formalize that $gcd(a, b)$ divides both a and b.

Hint: It is convenient to couch the developments in terms of the relation

$$\_\mathsf{divides}\_ : \forall\, (n\ m : \mathbb{N}) \to \mathsf{Set}$$

expressing the fact that n divides m, and to prove the lemma

$$\mathsf{divides}\dot{-} : \forall\, (n\ m\ k : \mathbb{N}) \to k\ \mathsf{divides}\ n \to k\ \mathsf{divides}\ m \to k\ \mathsf{divides}\ (m \mathbin{\dot{-}} n)$$

Exercise 3 Logarithms and Tree Heights (9 Points)

1. Implement the following functions calculating binary logarithms of natural numbers and rounding the result down and up respectively:

$$\lfloor \mathsf{log}_2\_ \rfloor : \mathbb{N} \to \mathbb{N}$$
$$\lceil \mathsf{log}_2\_ \rceil : \mathbb{N} \to \mathbb{N}$$

So, e.g. $\lfloor log_2 2 \rfloor = \lceil log_2 2 \rceil = 1$ and $\lfloor log_2 3 \rfloor = 1$, $\lceil log_2 3 \rceil = 2$. For the sake of simplicity, you can assume that $\lfloor log_2 0 \rfloor = \lceil log_2 0 \rceil = 0$.

2. Write down a function to calculate the hight of a brown tree:

   $\mathsf{bt\text{-}height} : \forall\ \{A : \mathsf{Set}\ \ell\}\{n : \mathbb{N}\} \to \mathsf{braun\text{-}tree}\ n \to \mathbb{N}$

3. Design a proof of the following property:

   $\mathsf{bt\text{-}height\text{-}lt} : \forall\ \{A : \mathsf{Set}\ \ell\}\{n : \mathbb{N}\} \to (t : \mathsf{braun\text{-}tree}\ n) \to (\mathsf{bt\text{-}height}\ t \leq \lceil \mathsf{log}_2\ n \rceil \equiv \mathsf{tt})$

To that end, you can use the following properties without a proof.

$\lfloor \mathsf{log}_2 \rfloor\mathsf{\text{-}dup} : \forall\ \{n : \mathbb{N}\} \to n > 0 \equiv \mathsf{tt} \to \lfloor \mathsf{log}_2\ (2\ *\ n) \rfloor \equiv \mathsf{suc}\ \lfloor \mathsf{log}_2\ n \rfloor$
$\lfloor \mathsf{log}_2 \rfloor\mathsf{\text{-}dup\text{-}suc} : \forall\ \{n : \mathbb{N}\} \to n > 0 \equiv \mathsf{tt} \to \lfloor \mathsf{log}_2\ (\mathsf{suc}\ (2\ *\ n)) \rfloor \equiv \mathsf{suc}\ \lfloor \mathsf{log}_2\ n \rfloor$