Travail de révision.

Exercice 1.1.

Montrer que si t1 et t2 sont bien formés, alors match t1 t2 est bien formé.

Cas 1 : vainqueur t1 ⩽ vainqueur t2

match t1 t2 = M (vainqueur t1) t1 t2.

vainqueur t1 est la plus petite valeur de t1

vainqueur t2 est la plus petite valeur de t2

Par (C1), on a que vainqueur t1 est la plus petite valeur de $[\![t]\!] = [\![t1]\!] \cup [\![t2]\!]$

Par hypothèse, t1 et t2 sont bien formés,

Donc, match t1 t2 est bien formé.

Cas 2 symétrique

Ce qui conclut la preuve.

Exercice 1.2

On veut montrer que $\overbrace{t \text{ est bien formé et } t \neq) k}^{(Pre)} \Rightarrow$

- k est la valeur minimale de $[\![t']\!]$, soit $k \in [\![t]\!]$ et $\forall n \in [\![t]\!], k \leq n$ (Q1)
- t' est bien formé et $[\![t']\!] = [\![t]\!] - \{|k|\}$ (Q2)
- hauteur t' ⩽ hauteur t (Q3)

avec (k, t') = delMin t.

On va montrer que delMin vérifie sa spécification par induction sur hauteur t.

On suppose que ∀u. hauteur u < hauteur t ∧ bienFormé(u) et u n'est pas de la forme ∫k, alors :

(H11) - K est la valeur minimale de ⟦u⟧, soit k∈⟦u⟧. et ∀n∈⟦u⟧, k≤n

(H12) - u' est bien formé et ⟦u'⟧ = ⟦u⟧ - {|k|}

(H13) - hauteur u' ≤ hauteur u

avec (k, u') = delMin u.

Cas 1 : t = (M k (∫ _) t')

delMin t = (k, t')

On sait que bienFormé(t) et que t ≠ ∫k par (C1) et (Rec), on a donc bienFormé(t') (Q1)

Comme k est le vainqueur de t1, alors t2 = ⟦t⟧ - {|k|} (Q2).

On sait que :

hauteur t = 1 + max (hauteur t1) (hauteur t2) > hauteur t' (Q3).

Cas 2 : t = M (_ t1 t2) avec t1 ≠ ∫k. (C2)

On a delMin t = (k, match t1' t2) en posant (k, t1') = delMin t1

On a hauteur t1 < hauteur t par définition de hauteur.

- Comme t est bien formé, on a que t1 est bien formé
- t1 ≠ ∫k (C2).

Comme le variant décroit et qu'il respecte les pré-conditions, on applique (HI).

Comme bienFormé(t) par (pre), on a vainqueur t = vainqueur t1

On a aussi que vainqueur t est la plus petite valeur de t et t1.

Comme k est la valeur la plus petite de t1, k est aussi la + petite valeur de t (Q1).

On a par (Pre) et (HI) que t1' et t2 sont bien formés.

Il vient avec Q1.1 que match t1' t2 est bien formé.

On a : $[\![\text{match } t1'\ t2]\!] = [\![t1']\!] \cup [\![t2]\!] = ([\![t1]\!] - \{|k|\}) \cup [\![t2]\!]$

$= [\![t1]\!] \cup [\![t2]\!] - \{|k|\} = [\![t]\!] - \{|k|\}$. (Q2)

hauteur (match t1' t2) = 1 + max (hauteur t1') (hauteur t2)

$\leq$ 1 + max (hauteur t1) (hauteur t2) (HI3)

$\leq$ hauteur t. (Q3)

Ce qui conclut la preuve.

Exercice 2.1.

```
joueurs :: [Int] -> [Tournoi]
joueurs [] = []
joueurs (a:l) = J a : joueurs l
```

Pré-condition aucune

Post-condition `ml l = mlt (joueurs l)`, `length l = length (joueurs l)`, tout élément de `tms(joueurs l)` est bien formé.

Variant `length l`

Montrez que `joueurs` vérifie sa spécification.

On souhaite montrer par induction sur length (que :

(Q1) • ml l = mlt (joueurs l)

(Q2) • length l = length (joueurs l)

(Q3) • tous les éléments de tms (joueurs l) bien formés.

On suppose (HI) que: $\forall l' < l \Rightarrow$

(HI1) : $ml\ l' = mlt\ (joueurs\ l')$

(HI2) : $length\ l' = length\ (joueurs\ l')$

(HI3) : tous les éléments de $trns\ (joueurs\ l')$ bien formés.

Cas 1 : $l = []$.

$joueurs\ l = []$.

On a (Q2) et (Q3) car $joueurs\ l = l$

On a (Q1) par vacuité.

Cas 2 : $l = a : l'$

$joueurs\ l =)\ a : (joueurs\ l')$

Comme $length\ l' < length\ l$, on applique (HI).

(HI1) : $ml\ l' = mlt\ (joueurs\ l')$

$\Rightarrow \{|a|\} \sqcup ml\ l' = \{|a|\} \sqcup mlt\ (joueurs\ l')$

$\Rightarrow ml\ l = mlt\ (joueurs\ l)$ (Q1)

(HI2) : $length\ l' = length\ (joueurs\ l')$

$\Rightarrow 1 + length\ l' = 1 + length\ (joueurs\ l')$

$\Rightarrow length\ (a : l') = length\ (a : joueurs\ l')$

$\Rightarrow length\ l = length\ (joueurs\ l)$ (Q2).

(HI3) tous les éléments de $trns\ (joueurs\ l')$ bien formés.

$\Rightarrow \{|a|\} \sqcup trns\ (joueurs\ l')$ bien formé $\Rightarrow trns\ (joueurs\ l)$ bien formé.

Exercice 2.2

Q 2.2 Jouer un tour dans un tournoi La fonction `joueTour` définie comme suit :

```
joueTour :: [Tournoi] -> [Tournoi]
joueTour []   = []
joueTour [t] = [t]
joueTour (t1:t2:l) = match t1 t2 : joueTour l
```

permet de combiner les tournois deux par deux. Nous spécifions l'appel `joueTour l` de la manière suivante :

Pré-condition pour tout `t` ∈ `tms l`, `t` est bien formé.

Post-condition `length(joueTour l) = length l ‘quot‘ 2 + length l ‘mod‘ 2, mlt(joueTour l) = mlt l` et tout `t` ∈ `tms(joueTour l)` est bien formé.

Démontrez que `joueTour` vérifie sa spécification.

On souhaite montrer que joueTour vérifie sa spécification par induction sur length l

On suppose (HI) que : ∀l'. length l' < length l ∩ ∀t∈tms l', bienFormé(t)

⇒

(HI1) - length (joueTour l') = length l' `quot` 2 + length l' `mod` 2.

(HI2) - mlt (joueTour l') = mlt l'.

(HI3) - ∀t'∈tms (joueTour l'), bienFormé(t')

Cas 2 : l = t1:t2:l' (C2)

joueTour l = (match t1 t2):(joueTour l')

Comme ⟦l'⟧ ⊂ ⟦l⟧, on a ∀t∈tms l, t bien formé

⇒ ∀t'∈tms l', t' bien formé

et length l' < length l, on applique (HI1) :

(HI3) : Comme ∀t∈tms l, bienFormé(t), on a bienFormé(ti) ∀i∈⟦1,2⟧

∀t'∈tms (joueTour l'), bienFormé(t')

⇒ ∀t∈{t1;t2}∪(tms (joueTour l')), bienFormé(t)

⇒ ∀t∈tms (joueTour l), bienFormé(t) (Q3)

(H12) : mlt (joueTour l') = mlt l'

=> ⟦t1⟧ ⊔ ⟦t2⟧ ⊔ mlt (joueTour l') = ⟦t1⟧ ⊔ ⟦t2⟧ ⊔ mlt l'

=> mlt (joueTour t1:t2:l') = mlt t1:t2:l' (Q2)

(H11) : length (joueTour l') = length l' `quot` 2 + length l' `mod` 2

(match t1 t2) : joueTour l' = joueTour l.

=> length (joueTour l) = 1 + length (joueTour l')

...

Q 2.3 Construire un tournoi unique Nous allons étudier maintenant la fonction `joueTournois` qui transforme une liste de tournois en un tournoi unique en s'appuyant sur la fonction `joueTour`.

```
joueTournois :: [Tournoi] -> Tournoi
joueTournois [t] = t
joueTournois l = joueTournois(joueTour l)
```

Voici la spécification pour l'appel `joueTournoi l` : (Pre1) (Pre2)

Pré-condition `l` $\neq []$ et tout élément de `tms l` est bien formé

Post-condition `joueTournoi l` est bien formé, $[\![$`joueTournoi l`$]\!] =$ `mlt l`.

Variant `length l`

Démontrez que `joueTournoi` satisfait sa spécification.

On veut montrer que joueTournois satisfait sa spécification par induction sur length l.

On suppose (HI) que ∀l'. length l' < length l ∧ l' ≠ [] ∧ tt élément de tms l est bien formé.

=> (HI1) • joueTournois l' est bien formé

(HI2) • mlt l' = ⟦ joueTournoi l' ⟧

Cas 1 : l = [t]

joueTournois l = t

Comme t est bienFormé (Pre2), joueTournois l est bien formé (Q1)

et ⟦ joueTournois l ⟧ = ⟦ t ⟧ = mlt l. (Q2)

Cas 2 : length l ⩾ 2 (C2)

joueTournois l = joueTournois (joueTour l)

Comme length joueTour l = length l `quot` 2 + length l `mod` 2

On a length l > length joueTour l

On a joueTour l ≠ [] et tt élément de mlt (joueTour l) est bien formé par Q22. On applique donc HI:

- (HI1): joueTournois (joueTour l) bien formé
  ⇒ joueTournois l bien formé (Q1)
- (HI2) mlt joueTour l = mlt l = ⟦joueTournois (joueTour l)⟧
  (Q2.2) = ⟦joueTournois l⟧ (Q2)

Ce qui conclut la preuve.

Q 2.4 D'un tournoi à une liste triée La fonction `tournoiListe` transforme un tournoi bien formé en une liste triée en éliminant les uns après les autres les vainqueurs des tournois avec la fonction `delMin`.

```
tournoiListe :: Tournoi -> [Int]
tournoiListe (J k) = [k]
tournoiListe t = let (k,t') = delMin t in k:tournoiListe t'
```

Voici la spécification de `tournoiListe t` :

Pré-condition `t` est bien formé

Post-condition `ml(tournoiListe t)` = ⟦t⟧ et `tournoiListe t` est triée.

Variant le nombre d'éléments de ⟦t⟧ (au sens des multi-ensembles, en comptant les multiplicités).

Démontrez que `tournoiListe` vérifie sa spécification.

On veut démontrer que tournoiListe vérifie sa spécification.

On suppose (HI) que : ∀t'. card ⟦t'⟧ < card ⟦t⟧ ∧ tournoi t' bien formé

⇒ (HI1) • ml (tournoiListe t') = ⟦t'⟧

(HI2) • tournoiListe t' triée

Cas 1: t = J k

tournoiListe t = [k] (Q2)

On a tournoiListe t triée et ml ([t]) = ⟦t⟧ (Q1)

Cas 2 : t = M k t1 t2

tournoiListe t = K : tournoiListe t'

t' est bien formé par Q1.2 car t est bien formé.

Comme card(⟦t'⟧) < card(⟦t⟧), on applique (HI):

(HI1) . ml(tournoiListe t') = ⟦t'⟧

=> {|k|} ⊔ ml(tournoiListe t') = ⟦t'⟧ ⊔ {|k|}

=> ml(k : tournoiListe t') = ⟦t⟧ => ml(tournoiListe t) = ⟦t⟧ (Q1)

(HI2) . tournoiListe t' est triée.

Comme k est la valeur minimale de ⟦t⟧, elle minore ⟦t'⟧

Donc, puisque tournoiListe t' est triée, il vient que k : tournoiListe t' est triée i.e. tournoiListe t est triée (Q2)

Ce qui conclut la preuve,

Q 2.5 Tri par tas-tournois

La fonction `triTournoi` construit un tournoi avec les éléments de la liste et retransforme ce tournoi en liste triée.

```
triTournoi :: [Int] -> [Int]
triTournoi [] = []
triTournoi l = tournoiListe (joueTournois (joueurs l))
```

Voici la spécification de `triTournoi l` :

Pré-condition aucune

Post-condition `ml l = ml(triTournoi l)` et `triTournoi l` est triée.

Démontrez que `triTournoi` vérifie sa spécification.

Cas 1 l = Σ. (C1)

triTournoi l = Σ. => (Q1) ∧ (Q2)

Cas 2 : l ≠ Σ.

triTournoi l = tournoiListe (joueTournois (joueurs l))

Tout élément de `joueurs` est bien formé (Q2.1)

Donc `joueTournois (joueurs ())` bien formé

et $[\![$ `joueTournois (joueurs ())` $]\!] =$ `mlt` $[\![$ `joueurs ()` $]\!]$ (Q2.3)

Donc, `tournoiListe (joueTournois (joueurs ()))` est bien formé et est trié (Q2.4).

Ce qui conclut la preuve.