Université de La Réunion mardi 5 juin 2001

DEUG 1^{ere} année – MIAS 1 et MASS 1

UE4 – Informatique 2 – Examen terminal

Durée : 2 heures – Aucun document n'est autorisé. 1

Répondre uniquement dans les cadres prévus à cet effet

Nom :	N° étudiant :	Signature
Prénom(s) :		
Date de naissance :		

Partie 1 : Listes préfixes

(2 pts) **Exercice 1 :** Écrivez une fonction `saufnpremiers` qui supprime les `n` premiers éléments d'une liste plate. Cette fonction renvoie la liste vide si `l` contient moins de `n` éléments. **Vous ne devez pas calculer la longueur de la liste.**

(3 pts) **Exercice 2 :** Écrivez un prédicat `prefixe?` qui prends en paramètres deux listes `l1` et `l2` et qui est vrai si la liste `l1` est préfixe de la liste `l2`.
Exemples : `(prefixe? '(1 2 3) '(1 2 3 d f g))` est vrai
`(prefixe? '(1 2 3) '(0 1 2 3 4 5))` est faux.

(5 pts) **Exercice 3 :** Écrivez une fonction `substitue` qui prend en paramètres trois listes `l1`, `l2` et `l` et qui substitue dans `l` toutes les apparitions de la liste `l2` par la liste `l3`. Vous pouvez utiliser la fonction prédéfinie `length`.
Exemple : `(substitue '(1 2) '(a b) '(5 1 2 x z 1 2 2))` renvoie `(5 a b x z a b 2)`

Université de La Réunion — mardi 5 juin 2001

DEUG 1^{ere} année – MIAS 1 et MASS 1

UE4 – Informatique 2 – Examen terminal

Durée : 2 heures – Aucun document n'est autorisé. **2**

Répondre uniquement dans les cadres prévus à cet effet

Nom :	N° étudiant :	Signature
Prénom(s) :		
Date de naissance :		

Partie 2 : Fonctions itératives

Exercice 1 : On donne la définition par récurrence de la suite de Fibonacci :

$$\begin{array}{lcll} \text{fib}: & \textit{Entier} & \longrightarrow & \textit{Entier} \\ & n & \longmapsto & \begin{cases} 0 & \text{si } n{=}0 \\ 1 & \text{si } n{=}1 \\ fib(n-2)+fib(n-1) & \text{si } n \geq 2 \end{cases} \end{array}$$

(2 pts) Écrivez en Scheme la fonction **fiborec** (non récursive terminale) directement inspirée de la définition par récurrence ci-dessus.

(4 pts) Écrivez en Scheme **une version itérative** avec paramètres accumulateurs de la fonction précédente.

Tournez SVP $\Longrightarrow$

(4 pts) **Exercice 2 :** Écrivez **de manière itérative** une fonction **ote** qui prend comme paramètres un objet o et une liste plate l et qui supprime toutes les occurences de o dans l.

Université de La Réunion — mardi 5 juin 2001

DEUG 1^{ere} année – MIAS 1 et MASS 1

UE4 – Informatique 2 – Examen terminal

Durée : 2 heures – Aucun document n'est autorisé. **3**

Répondre uniquement dans les cadres prévus à cet effet

Nom :	N° étudiant :	Signature
Prénom(s) :		
Date de naissance :		

Partie 3 : Les Files

Une file est un type abstrait de données linéaire. Les éléments sont rangés les uns à la suite des autres ; deux éléments sont privilégiés : le premier ou tête de file qui est l'élément supprimé lors d'un retrait dans la file, et le dernier ou queue de file qui est le dernier élément ajouté. Voici un exemple de file d'entiers : `-> 12 6 9 -7 2 ->` Les flèches sont là pour donner le sens de la file, i.e. la tête (ici 2) et la queue (ici 12).

Ci-dessous sont présentées les spécifications des fonctions caractéristiques des files telles que les fournit l'auteur des dites fonctions. **Attention, il ne vous est pas demandé de coder ces fonctions !**

`file_vide` : $\rightarrow$ *file*
$\mapsto$ la file vide

`ajout` : *Element* $\times$ *file* $\rightarrow$ *file*
e, f $\mapsto$ ajoute l'élément e en queue de la file f

`retrait` : *file* $\rightarrow$ *file*
f $\mapsto$ la file f privée de son premier élément

`premier` : *file* $\rightarrow$ *Element*
f $\mapsto$ l'élément de tête de f

`est_vide?` : *file* $\rightarrow$ *booleen*
f $\mapsto$ `#t` ssi f est la file vide

(2 pts) **Question 1 :** Écrivez en Scheme une fonction **longueur** qui renvoie le nombre d'éléments d'une file donnée.

Tournez SVP $\Longrightarrow$

Question 2 : Écrivez en Scheme une fonction **retourne** qui retourne les éléments d'une file. Par exemple `retourne` *(2 pts)* appliquée à la file `-> 12 6 9 -7 2 ->` renvoie `-> 2 -7 9 6 12 ->`

Question 3 : Écrivez en Scheme une fonction **place** qui, pour une file f et une élément e donnés, renvoie le numéro de e dans f ou -1 si e n'est pas dans f. Par exemple, pour la file `-> 2 -7 9 6 12 ->` et l'élément `-7`, `place` *(3 pts)* renvoie 4.

Question 4 : Écrivez en Scheme une fonction **fusionne** qui met bout à bout deux files f_1 et f_2 pour former une seule file. Par exemple, `fusionne` sur les files `-> 17 5 -3 1 ->` et `-> 4 9 ->` renvoie la file `-> 17 5 -3 1 4 9 ->`. *(3 pts)*

Université de La Réunion — mardi 5 juin 2001

DEUG 1^{ere} année – MIAS 1 et MASS 1

UE4 – Informatique 2 – Examen terminal

Durée : 2 heures – Aucun document n'est autorisé. **4**

Répondre uniquement dans les cadres prévus à cet effet

Nom :	N° étudiant :	Signature
Prénom(s) :		
Date de naissance :		

Partie 4: Ordre supérieur

(2 pts) **Exercice 1:** Pour tout $n \in \mathbb{N}^*$, on note f_n la fonction de $\mathbb{N}$ dans $\mathbb{N}$ définie par

$$f_n \ : \ m \mapsto \begin{cases} 0 & \text{si m multiple de } n \text{ et } m \neq n \\ m & \text{sinon} \end{cases}$$

Écrivez en Scheme la fonction **calculer-fn** qui prend en argument un entier naturel non nul n et renvoie la fonction f_n.

(2 pts) **Exercice 2:** Écrivez en Scheme la fonction **supprimer** qui, étant donnés $n \in \mathbb{N}^*$ et une liste d'entiers l, remplace par 0 tous les éléments de l à la fois distincts de n et multiples de n. Les autres éléments de l doivent rester inchangés. Vous utiliserez les fonctions **calculer-fn** et **map**.

Exemple : `(supprimer 5 '(1 5 8 10 9 15))` doit renvoyer la liste `(1 5 8 0 9 0)`.

Tournez SVP $\Longrightarrow$

(3 pts) **Exercice 3:** On suppose donnée la fonction Scheme **liste1-a-n** qui, étant donné $n \in \mathbb{N}^*$, renvoie la liste des entiers allant de 1 à n. En utilisant **liste1-a-n** et **supprimer**, écrivez en Scheme la fonction **premiers0** qui, étant donné $n \in \mathbb{N}^*$, renvoie la liste où tous les nombres compris entre 1 et n ont été remplacés par 0, sauf ceux qui sont premiers.

Exemple : `(premiers0 7)` doit renvoyer la liste `(1 2 3 0 5 0 7)` (4 et 6 n'étant pas premiers, ils sont remplacés par 0).

(3 pts) **Exercice 4 :** Écrivez en Scheme la fonction **premiers** qui, étant donné $n \in \mathbb{N}^*$, renvoie la liste des nombre premiers compris entre 1 et n. Vous utiliserez les fonctions **premiers0**, **apply** et **map** ainsi que l'astuce suivante (qui reprend l'exemple précédent).

Astuce: `(append '(1) '(2) '(3) '() '(5) '() '(7))` renvoie la liste `(1 2 3 5 7)`.

Remarque: il s'agit juste en fait de supprimer les 0 de la liste renvoyée par la fonction **premiers0**!