Université Paris Diderot
M1 Informatique

Programmation Fonctionnelle Avancée
Année 2013-2014

TP n°8

Structures de données fonctionnelles efficaces

Exercice 1 Implémentez la signature DEQUE donnée dans les transparents du cours (sans les fonctions `head` et `last`).

Procédez ensuite à une analyse de coût amorti.

Exercice 2 Implémentez les fonctions suivantes sur les streams :

```
module type STREAMEXT = sig
  include module type of Stream
  val of_list : 'a list -> 'a stream
  val to_list : 'a stream -> 'a list
  val push : 'a -> 'a stream -> 'a stream
  val pop : 'a stream -> ('a * 'a stream) option
  val map : ('a -> 'b) -> 'a stream  -> 'b stream
  val filter : ('a -> bool) -> 'a stream -> 'a stream
  val split : 'a stream -> 'a stream * 'a stream
  val join : 'a stream * 'a stream -> 'a stream
end;;
```

Le code source est disponible à l'adresse :
`http://gallium.inria.fr/~scherer/teaching/fpav/2013/tp8/exo_stream.ml`

Exercice 3 En utilisant la fonction `Unix.gettimeofday`, comparez le coût des trois différentes files vues en cours de différentes manières (pour un usage non persistant). Pensez à supprimer tous les appels au module `Printf` pour ne pas fausser les résultats.

- En ne faisant qu'ajouter des éléments : mettez en évidence le surcoût des structures paresseuses tant au niveau mémoire que processeur.
- En faisant des ajouts et des suppressions pour tester le coût de remove. En particulier le coût "constant" de add est-il le même que le coût "constant" de remove ?
- Grâce à la commande `gnuplot`, tracez les graphes (temps en fonction des opérations) pour les deux files avec stream pour lesquels vous insérez 10.000.000 d'éléments, puis vous les supprimez tous. Vous devrez jouer sur la précision (tous les combiens d'opérations vous mesurez le temps). Pour bien observer les différences, mettez les fonctions sur le même graphe.