DS 0

Option informatique, deuxième année

Julien Reichert

Durée : environ une heure et demie

Exercice 1 : Écrire en Caml une fonction `retrait ch l` qui crée une chaîne de caractères obtenue en retirant de `ch` des caractères comme précisé dans `l`, qui est une liste de couples chacun formé par un caractère `c` et un entier naturel `n`, de sorte que les `n` premières occurrences de `c` soient à retirer.

Par exemple, l'appel `retrait "Bonjour le monde" [('o', 2); ('n', 3)]` produira `"Bjur le mode"`, sans erreur même si `'n'` n'apparaît que deux fois.

Exercice 2 : Écrire en Caml une fonction `decoupage long larg` qui détermine la liste des côtés des carrés permettant de paver un rectangle dont les dimensions sont en argument de la fonction. Il s'agit d'optimiser cette liste en prenant à chaque étape le plus grand côté possible pour le carré. En particulier, la liste sera décroissante.

Exercice 2bis : Déterminer la complexité temporelle de la fonction précédente dans le pire des cas avec le plus de rigueur possible (ce n'est pas forcément évident).

Exercice 3 : Écrire en Caml une fonction `cordes l`, résolvant le problème suivant : on dispose de cordes dont les longueurs sont données dans la liste `l`, et on les réduit étape par étape en retirant à chaque corde la longueur de la plus petite, faisant disparaître celle-ci ainsi que celles qui avaient la même taille ; il s'agit de déterminer à chaque étape combien il reste de cordes en tant que liste d'entiers, dont le `0` final n'est pas inclus car il est automatique.

Par exemple, l'appel `cordes [3; 3; 2; 9; 7]` retournera `[5; 4; 2; 1]`, car au début il y a cinq cordes, puis quand chacune aura été réduite de deux unités il en restera quatre, puis quand celles-ci auront été réduites d'une unité les deux premières de la liste disparaîtront, etc.

Exercice 3bis : Déterminer la complexité temporelle de la fonction précédente dans le pire des cas. Discuter de la possibilité de changer d'algorithme pour rendre la complexité moindre ou améliorer la simplicité d'écriture (suivant le cas).

Exercice 4 : Écrire en Caml une fonction `temps l n` qui détermine le temps total d'attente à une caisse de supermarché, en fonction de la liste `l` des temps mis par chaque client et du nombre `n` de caisses. Le principe est que chaque client dans l'ordre aille à une caisse libre dès qu'il y en a une, et qu'on détermine à quel moment tous les clients auront terminé.

Par exemple, l'appel `temps [2; 3; 10] 2` retournera `12`, car les deux premiers clients occuperont les caisses et le troisième devra attendre que le premier ait terminé. On observe ainsi que ce n'est pas l'ordre optimal.

Exercice 5 : Écrire en Caml une fonction `flavius n k` qui détermine l'ordre d'élimination des nombres de `1` à `n` en s'inspirant de l'histoire de Flavius Josèphe : la liste des nombres de `1` à `n` est parcourue de manière circulaire, et chaque fois qu'on a compté `k` éléments, l'élément en cours de parcours est éliminé.

Par exemple, pour `n` valant `7` et `k` valant `3`, on élimine `3` puis `6`, en revenant au début après avoir parcouru `7` on élimine `2`, puis `7`, puis `5`, puis `1` (parcouru deux fois) et finalement `4`, qui était le dernier élément. Cela donne finalement la liste `[3, 6, 2, 7, 5, 1, 4]`.