DS 2

Informatique MP2I

Julien Reichert

Toutes les questions de programmation sont à résoudre (on s'en doute) en OCaml.

Questions de cours ou d'application directe du cours

Question de cours 1 : Donner au moins trois informations contenues dans un nœud d'index d'après le cours.

Question de cours 2 : Donner une différence importante entre un lien physique et un lien symbolique.

Question de cours 3 : Donner un exemple d'utilisation de la commande `chmod` dans l'une des deux versions au choix de la syntaxe, et expliquer l'effet de la commande.

Question de cours 4 : Quelle est l'utilité des assertions et pourquoi évite-t-on de s'en servir en pratique ?

Question de cours 5 : Définir la notion de graphe de flot de contrôle.

Question de cours 6 : Donner le principe du calcul de complexité d'une fonction récursive, avec un petit exemple (ne pas dépasser une demi-douzaine de lignes, ce serait inutile).

Question de cours 7 : Donner les opérateurs en OCaml pour les tests d'égalité et de différence.

Question de cours 8 : Comment accède-t-on au premier caractère d'une chaîne nommée `s` (on suppose que la chaîne n'est pas vide) ?

Question de cours 9 : Quel est le problème dans le code suivant, écrit en OCaml ? (On ne demande pas de corriger le problème !)

```
let liste_entiers n =
  match n with
  | 0 -> []
  | _ -> let l = ref [] in
    for i = 1 to n do i :: !l done;
    !l
```

Exercices issus des TD et TP

Exercice T1 : Déterminer comment obtenir récursivement l'ensemble des arrangements de `p` éléments d'un ensemble à `n` éléments. Les répétitions sont ici impossibles, en particulier on suppose que `p` est entre `0` et `n`. Réaliser ceci sous la forme d'une fonction en OCaml.

Exercice T2 : Écrire une fonction récursive qui prend en entrée une liste de flottants et qui retourne le produit de ses éléments.

Exercice T3 : Écrire une fonction récursive qui prend en entrée une liste et qui détermine si ses éléments sont dans l'ordre croissant ou dans l'ordre décroissant (auquel cas la réponse est `true`, et dans tous les autres cas la réponse est `false`).

Exercice T4 : Écrire une fonction qui prend en entrée deux références et qui échange leur contenu. Attention, il y a un piège, donc mieux vaut s'assurer que la fonction marche. Anticiper la signature avant de la consulter.

Exercices

Exercice 1 : Écrire une fonction prenant en argument un tableau (qu'on supposera non vide) et renvoyant un indice de son maximum (en cas d'égalité, n'importe lequel pourra être choisi).

Exercice 2 : Écrire une fonction prenant en argument deux chaînes de caractères ainsi qu'un caractère et déterminant si le caractère en question apparaît strictement plus souvent dans la première chaîne ou non.

Exercice 3 : Écrire une fonction prenant en argument un entier positif (pas à vérifier) noté `n` et renvoyant le plus grand entier qui soit inférieur ou égal à `n` et qui soit par ailleurs le carré ou le cube d'un entier.

Exercice 4 : Écrire une fonction prenant en argument un tableau ainsi qu'une valeur du même type que les éléments du tableau et renvoyant le nombre d'indices minimal entre deux occurrences de la valeur en question (on renverra `-1` si la valeur n'apparaît pas au moins deux fois dans le tableau, et on renverra en particulier la différence des indices s'il y a exactement deux occurrences).

Exercice 5 : Montrer que la fonction précédente termine (en admettant que ce ne soit quand même pas trivial) et qu'elle est correcte. Bien utiliser les termes officiels du cours dans la réponse, non évaluée s'ils n'y figurent pas.

Exercice 6 : Écrire une fonction prenant en argument une liste de couples de flottants (le premier élément de chaque couple étant inférieur au second, propriété qui n'est pas à vérifier) et renvoyant l'écart minimal entre deux éléments d'un couple. Si la liste est vide, on renverra une erreur.

Exercice 7 : Écrire une fonction prenant en argument une liste et renvoyant le nombre de « phases de monotonie » dans la liste, en définissant une phase de monotonie comme une sous-liste monotone d'éléments consécutifs de la liste qu'on ne peut pas prolonger sans perdre la monotonie (ou parce qu'on est à un bout de la liste). Attention, la liste elle-même pourra avoir des éléments consécutifs égaux !

Problème : Stéganographie

La stéganographie consiste à cacher une information dans des données d'apparence anodine. Elle a eu de nombreuses illustrations dans l'histoire et fait partie des domaines qui permet de découvrir l'informatique de manière divertissante.

Une fois n'est pas coutume, il y aura du cours (hors programme certes, mais au programme du tronc commun) dans un énoncé de DS, donc il est recommandé de lire l'intégralité du problème, quitte à ce que ce soit après le devoir.

Ce problème aborde un cas d'application simple de la stéganographie, en cachant un texte (ou une petite image) dans une image.

Certes, la stéganographie est bien plus complexe que ce que l'on verra ici, mais il y a eu des TIPE sur le sujet qui n'allaient pas forcément bien plus loin du point vue de la programmation seule...

Aucune connaissance préalable du traitement des images n'est supposée, les informations utiles seront fournies.

Question P1 : Une image est assimilée à une matrice de triplets d'entiers entre `0` et `255`, les triplets signalant le degré d'intensité des trois lumières primaires (dans l'ordre les lumières rouge, verte et bleue). Ceci permet d'attribuer la couleur d'un carré élémentaire de l'image appelé pixel (le nom vient du terme anglais *picture element*). Sans compression, quel est alors le poids en octets d'une image en fonction du nombre de lignes et du nombre de colonnes ?

Question P2 : En pratique, les images sont compressées (le format `bmp` ne l'est pas forcément, mais les photos utilisant le format `jpg`, et les images engendrées par l'ordinateur utilisant le format `png` le sont, le format `png` n'occasionnant par ailleurs pas de perte de données). Quel serait alors le rapport taille après compression sur taille originale pour une image de hauteur 2000, de largeur 4000 et stockée dans un fichier de 1858 kilooctets (avec la définition commerciale du kilooctet, c'est-à-dire 10^3 octets) ?

Remarque importante : On ignorera totalement la notion de compression pour la suite du sujet.

À l'œil nu, on peut estimer qu'une différence sur les quatre bits de poids faible de chaque teinte ne peut pas être distinguée (cela dépend certes des individus), ainsi le triplet `(64, 16, 128)` pourra être confondu avec le triplet `(67, 21, 131)` et le représentera en tant que triplet où les quatre bits de poids faible de chaque teinte ont été mis à zéro.

Question P3 : Écrire une fonction prenant en argument un entier (qu'on pourra considérer comme étant entre `0` et `255`, mais ce n'est pas à vérifier) et renvoyant son représentant, c'est-à-dire l'entier dont la représentation binaire a les mêmes bits sauf les quatre de poids faible qui sont mis à zéro.

En considérant des données à cacher dans l'image en tant que séquence de bits, l'opération revient à écrire, ligne par ligne, colonne par colonne et teinte par teinte (dans l'ordre rouge puis vert puis bleu), les bits de la séquence à la place des bits de poids faible des teintes, en mettant en particulier le premier bit de la séquence dans le bit correspondant à 2^3 de la teinte rouge du pixel à la première colonne de la première ligne.

Question P4 : Si la séquence de bits à cacher correspond à un fichier texte, quelle taille de fichier peut être cachée dans une image de hauteur 2000 et de largeur 4000 ?

Question P5 : Si la séquence de bits à cacher correspond à une image de même ratio (c'est-à-dire le rapport largeur sur hauteur) que l'image, quelle inégalité les dimensions de l'image à cacher et celles de l'image de départ vérifient-elles si on ne veut pas « flouter » l'image à cacher ? (Il est vrai qu'en pratique les bits de poids faible peuvent être perdus dans l'image à cacher de sorte que les dimensions coïncident dans certains cas d'application…)

Question P6 : Écrire une fonction prenant en argument une matrice de triplets d'entiers correspondant à une image et un tableau de booléens correspondant à la séquence de bits à cacher et renvoyant une nouvelle matrice de triplets d'entiers correspondant à l'image avec la séquence de bits cachée dans les bits de poids faible selon les principes ci-avant. Si l'image est trop petite pour tout contenir, on déclenchera une erreur. Les pixels restants de l'image seront laissés intacts et le dernier pixel impacté verra toutes les teintes modifiées quitte à mettre des zéros additionnels.

Question P7 : Écrire la fonction réciproque, mais il s'agit de rassembler les booléens par huit et de produire une chaîne de caractères à partir des octets ainsi séparés. Si le nombre d'octets n'est pas entier, on complètera le dernier avec des zéros. On signale ou rappelle que la fonction à utiliser en OCaml est `char_of_int`. On rappelle que la chaîne représente un fichier donc le caractère zéro en marque la fin.

Question P8 : En admettant que la fonction précédente ait été incorporée dans un programme qui ouvre une image dont le chemin est spécifié dans une variable globale (qu'on n'aura pas à toucher), extrait son contenu dans une matrice de triplets d'entiers, appelle la fonction susmentionnée et imprime son résultat, et que le programme en question ait été compilé pour donner l'exécutable qu'on nommera `g` (sans argument, eu égard à la variable globale en question) et qui est dans le répertoire courant, écrire une ligne de commande pour un terminal Unix qui appelle l'exécutable et qui stocke le résultat dans un fichier qu'on appellera `p.txt`.

La (brève) consigne de cette huitième question a été grandement simplifiée puisque la notion d'argument de ligne de commande n'a pas encore été vue…

Question P9 : En parlant de stéganographie, question bonus : quel célèbre compositeur d'opéras est décédé il y a cent ans jour pour jour en ce 29 novembre ? Des indices pour le retrouver sont disséminés dans le sujet.

Un petit logimage / nonogram / picross pour finir (toujours pas noté, mais cela peut rapporter des points quand même...).

Si les règles ne sont pas connues (bien triste vu le divertissement que ce jeu apporte) : il s'agit de noircir certaines cases de la grille pour former un dessin. Devant chaque ligne et chaque colonne figurent des indices révélant le nombre de cases noires consécutives de chaque bloc de cases noires dans l'ordre, sachant que les blocs sont séparés d'au moins une case blanche. Une ligne ou colonne sans indice n'a en particulier que des cases blanches.