DS 2

Informatique MP2I

Julien Reichert

Questions de cours

Question de cours A1 : Comment prouver la terminaison d'une boucle conditionnelle ?

Question de cours A2 : Comment calculer la complexité d'une fonction récursive ?

Question de cours A3 : Expliquer brièvement (inutile de dépasser quatre lignes) le principe de la représentation des flottants.

Questions de cours en OCaml

Question de cours B1 : Écrire une fonction qui prend en entrée une chaîne de caractères et qui détermine son dixième caractère. Au préalable, on devra vérifier que la taille est suffisante et si ce n'est pas le cas le comportement est au choix entre déclencher une erreur et retourner le caractère de fin de chaîne.

Question de cours B2 : Écrire un morceau de code qui crée un type s'apparentant aux listes d'entiers. Le constructeur de liste vide sera `V` et le constructeur récursif sera `C`, paramétré par un entier et une liste d'entiers du type en construction.

Question de cours B3 : Considérons le type suivant : `type t1 = { a : int ; b : string ; c : int}`. Créer une variable de type `t1`. Quel mot-clé doit être ajouté à la création du type pour permettre de rendre l'attribut `b` mutable ? Dans ce cas, écrire un morceau de code qui mute cet attribut pour la variable créée en lui affectant une autre chaîne au choix.

Questions de cours en C

Question de cours C1 : Parmi les types suivants, le(s)quel(s) n'existe(nt) pas : `unsigned int`, `float`, `string`, `double` ?

Question de cours C2 : Donner les trois opérateurs booléens usuels.

Question de cours C3 : Soient deux tableaux `t1` et `t2` de même type et de même taille `n`, toutes ces valeurs correspondant à des variables déclarées. Écrire un morceau de code pour que le contenu de `t1` devienne le même que celui de `t2` sans lier les deux tableaux.

Exercices en OCaml

Exercice 1 : Soit la fonction `let c = fun g f -> (fun x -> g (f x));;`. Donner le type de `c` et un exemple d'utilisation, résultat de l'évaluation compris.

Exercice 2 : Écrire une fonction prenant en argument un tableau de tableaux de booléens, tous de même taille non nulle (qui n'est cependant pas forcément égale au nombre de tableaux, lui aussi non nul par ailleurs) et retournant le tableau obtenu en prenant la conjonction de toutes les "colonnes" du tableau en entrée, c'est-à-dire que l'élément d'indice `i` du résultat sera vrai si, et seulement si, les booléens à l'indice `i` de tous les tableaux du tableau en argument sont vrais. On ne vérifiera ni que les tailles sont toutes les mêmes, ni qu'elles sont différentes de zéro, ni qu'il y a au moins un tableau, ce sera une garantie.

Exercice 3 : Écrire une fonction qui prend en entrée un tableau de type quelconque et un prédicat prenant en argument des valeurs du type en question et qui retourne la position de départ et la longueur de la plus longue suite d'indices consécutifs de valeurs vérifiant le prédicat.

Exercices en C

Exercice 4 : Écrire une fonction qui prend en argument deux entiers et qui retourne leur PGCD. L'algorithme sera nécessairement écrit, malgré le manque d'optimalité, sur la base de tests de divisibilité commune.

Exercice 5 : Écrire une fonction qui prend en argument un tableau d'entiers et sa taille et qui retourne la médiane du tableau. On supposera la taille impaire de sorte qu'il n'y ait qu'un élément médian. La complexité peut être quadratique mais on ne mutera pas le tableau.

Exercice 6 : Expliquer ce que fait la fonction suivante :

```
int *f(int n, int tailleres)
{
  int *p = malloc(tailleres*sizeof(int));
  for (int i = tailleres-1 ; i >= 0 ; i -= 1)
  {
    p[i] = n % 2;
    n /= 2;
  }
  return p;
}
```

Problème

Ce problème s'intéresse à la manipulation d'entiers dépassant la taille prévue pour le type idoine, en utilisant soit des chaînes de caractères (pas standard) soit des tableaux d'entiers dont le premier signale le nombre d'entiers manipulés (standard). **La première technique sera traitée intégralement en C, la deuxième intégralement en OCaml.**

Pour les questions difficiles, l'utilisation de fonctions annexes sera bienvenue, même si aucun énoncé ne le suggère.

Arithmétique des chaînes de caractères

Considérons des chaînes de caractères en C dont les éléments seront obligatoirement des caractères correspondant à un chiffre. Les entiers ainsi manipulés seront alors par principe positifs, pour une modélisation simplifiée.

Les opérations seront effectuées caractère par caractère, en propageant l'éventuelle retenue.

Tous les calculs produiront une nouvelle chaîne de caractères dans laquelle le résultat sera stocké. **Cependant, toute consommation excessive de mémoire sans recyclage sera sanctionné.**

Le chiffre des unités sera mis dans le premier caractère de la chaîne, afin que des tailles possiblement différentes n'occasionnent pas de difficultés d'alignement.

Question P1 : Pourquoi avoir choisi des chaînes de caractères plutôt que des tableaux de nombres à un chiffre ? (Plusieurs réponses pertinentes sont possibles, on se contentera d'en donner une.)

En pratique, on n'utilisera pas le binaire pour éviter la redondance avec la section suivante et pour adopter le point de vue de « manipuler les données que l'utilisateur voit effectivement ».

Question P2 : Écrire une fonction qui prend en argument une chaîne de caractères représentant un entier selon la méthode de cette section et qui retourne l'entier représenté. On supposera sans le vérifier que l'argument est une chaîne représentant un entier tenant sur 31 bits, de sorte qu'aucun dépassement ne risque de se produire.

La fonction `atoi` peut servir pour la question précédente...

Question P3 : Écrire une fonction qui prend en argument une chaîne de caractères `s` représentant un entier selon la méthode de cette section ainsi qu'un entier `i` et qui retourne une chaîne représentant le même entier où une retenue est ajoutée à la position `i`, sachant que cette retenue peut être amenée à se propager.

On rappelle que pour qu'un tableau créé dans une fonction survive à celle-ci, une allocation de mémoire est nécessaire...

Question P4 : Écrire une fonction qui prend en argument deux chaînes de caractères représentant des entiers selon la méthode de cette section et qui retourne une chaîne représentant leur somme.

Question P5 : Écrire une fonction qui prend en argument deux chaînes de caractères représentant des entiers selon la méthode de cette section et qui retourne une chaîne représentant leur produit. On ne fera pas la somme en boucle, dans la mesure où d'une part la complexité serait excessive et d'autre part le seuil de la boucle ne pourrait a priori pas être représenté par une variable de type entier par exemple.

Arithmétique des tableaux d'entiers

Considérons des tableaux d'entiers (sur 64 bits) en OCaml servant à représenter des entiers relatifs selon le principe suivant : on découpe l'entier en somme de puissances de 2^{31}, la plus forte puissance portant le signe de l'entier, et on écrit chaque coefficient dans un tableau, la plus forte puissance à l'indice 0.

Ainsi, le nombre qui s'obtient par l'addition $2^{100}+2^{50}+2^{25}+1$ se découpe en $2^7\times2^{93}+0\times2^{62}+2^{19}\times2^{31}+(2^{25}+1)\times2^0$ et se représente dans le tableau `[|128; 0; 524288; 33554433|]`.

Attention, le nombre qui s'obtient par l'addition $-2^{42}+2^5$ se découpe en $-2^{11}\times2^{31}+2^5\times2^0$, ce qui est plus agréable à représenter que son écriture binaire qui comporterait un signe moins et une trentaine de chiffres `1`. On a ici le tableau `[|-2048; 32|]`.

Question P6 : Pourquoi avoir choisi des puissances de 2^{31} et non pas 2^{64} par exemple ?

Question P7 : Écrire une fonction qui prend en argument deux tableaux d'entiers représentant des entiers selon la méthode de cette section et qui retourne un tableau d'entiers représentant leur somme.

Question P8 : Écrire une fonction qui prend en argument deux tableaux d'entiers représentant des entiers selon la méthode de cette section et qui retourne un tableau d'entiers représentant leur produit. On ne fera pas la somme en boucle, pour limiter la complexité. **Pour éviter que la difficulté ne soit trop élevée, les nombres seront forcément positifs.**

En pratique, des algorithmes similaires à ceux qu'on retrouve pour les calculs rapides concernant les polynômes (Karatsuba, voire transformée de Fourier) sont employés lorsqu'une représentation d'entiers de précision arbitraire est rencontrée...

Désormais, la méthode de représentation change, et on découpe l'entier en somme de puissances d'un milliard.

Question P9 : Écrire une fonction qui prend en argument un tableau d'entiers représentant un entier selon la nouvelle méthode et qui imprime l'entier représenté.

Un peu d'aide : Pour imprimer un entier de sorte à ce qu'au moins neuf caractères figurent, dont des zéros pour compléter si nécessaire, on peut utiliser `Printf.printf "%09d" n` où `n` est l'entier.

Question P10 : Proposer une méthode pour imprimer un entier représenté avec la méthode précédente, éventuellement à la lumière de cette nouvelle méthode. (Plusieurs réponses pertinentes sont possibles.)