DS 1

Informatique MP2I

Julien Reichert

Toutes les questions de programmation sont à résoudre en OCaml.

Question de cours 1 : Quelle fonction donne la taille d'une liste et quelle est sa complexité ?

Question de cours 2 : Donner au moins une fonction permettant de créer un tableau et donner sa spécification.

Question de cours 3 : Donner la définition d'un type somme et d'un type enregistrement, exemples à l'appui.

Question de cours 4 : Rappeler la différence entre `::` et `@` et préciser le type de leurs opérandes.

Question de cours 5 : Comment mute-t-on une référence (donner un exemple) ?

Exercice 1 [Programmation impérative attendue en vue de la rédaction des preuves!] : Écrire une fonction de signature `premier_libre_depuis : int array -> int -> int` qui prend en argument un tableau d'entiers `tab` et un entier `n` et qui détermine la plus petite valeur supérieure ou égale à `n` qui soit absente du tableau. Prouver la terminaison et la correction de la fonction. Il n'y a pas d'objectif de complexité (dans la limite de la décence) dans cette question, ni de bonus si la complexité est particulièrement basse, mais **la complexité doit être calculée à la suite de l'écriture de la fonction**.

Exercice 2 [Programmation impérative attendue en vue de la rédaction des preuves!] : En reprenant l'exercice précédent, on suppose que le tableau est **strictement** croissant (sans le vérifier). Il est alors possible d'écrire une fonction dont la complexité est logarithmique dans tous les cas. Écrire une nouvelle version de la fonction attendue ayant cette complexité. Prouver la terminaison et la correction de la fonction. Le calcul de complexité n'est pas nécessaire.

Exercice 3 : Écrire une fonction `moyminmax` prenant en argument une liste de nombres (au choix entiers ou flottants) et renvoyant la moyenne arithmétique **sans arrondi** entre le minimum et le maximum de la liste.

Exercice 4 : Sans utiliser les fonctions `of_list` et `to_list` du module `Array`, écrire deux fonctions de même effet (conversion d'une liste en le tableau des éléments dans le même ordre et vice-versa).

Exercice 5 : Écrire une fonction de signature `nthnth : 'a list list -> int -> int -> 'a` de sorte que `nthnth listeliste i j` renvoie l'élément qui serait d'indice `j` dans la liste qui serait d'indice `i` de `listeliste`... si les listes étaient indexables, bien entendu. Le comportement est au choix en cas de débordement de `i` ou `j`. **L'utilisation de la fonction `List.nth` est ici interdite.**

Exercice 6 (difficile) : L'extraction de sous-tableaux en OCaml est limitée : les éléments sont forcément consécutifs, et déborder mène à une erreur. Écrire une fonction de signature `slice : 'a array -> int -> int -> int -> 'a array` telle que `slice tab d f p` retourne un sous-tableau du tableau `tab` formé des éléments aux positions de `d` inclus à `f` exclu par pas de `p`, avec les conditions suivantes : on impose que `d` soit compris entre `0` et la taille de `tab` moins un, que `f` soit compris entre `-1` (pour pouvoir aller jusqu'au début dans le cas d'un pas négatif) et la taille de `tab` (pour pouvoir aller jusqu'à la fin dans le cas d'un pas positif), et que `p` soit non nul, tous les arguments sont obligatoires et le résultat est vide si le pas ne va pas dans le bon sens par rapport aux seuils. Ainsi, on imite une version restreinte de `tab[d:f:p]` de Python.

Problème 1 : Manipulations sur la représentation des nombres.

Le but de ce problème est de recréer la représentation des flottants en stockant soi-même les bits dans un tableau de booléens. *On ne tentera pas de dire qu'il suffit de recopier le contenu de la mémoire ou toute arnaque de ce genre !*

On rappelle que selon la norme IEEE754, un flottant représenté sur 64 bits s'écrit comme le bit de signe puis onze bits d'exposant puis cinquante-deux bits de mantisse, sachant que les bits d'exposant correspondent à l'encodage en tant qu'entier positif de la puissance de deux du nombre en écriture scientifique binaire augmentée de 1023, en admettant que cette puissance soit comprise entre -1022 et 1023 inclus, ce qui laisse deux valeurs de l'exposant pour représenter des flottants spéciaux.

La mantisse, quant à elle, correspond aux cinquante-deux premiers chiffres binaires après la virgule, toujours en écriture scientifique binaire et en admettant que l'exposant soit dans l'intervalle autorisé.

Question P1 : Rappeler aussi à partir du cours comment on représente une valeur nulle et écrire une fonction de signature `zero_b : bool array -> bool` déterminant si un tableau de booléens représente une valeur nulle.

Question P2 : Écrire une fonction de signature `denor_b : bool array -> bool` déterminant si un tableau de booléens représente une valeur spéciale dans le sens où une des deux valeurs exclues de l'exposant est utilisée.

Question P3 : Donner la représentation (points de suspension autorisés) du plus grand flottant hors valeurs spéciales et celle du plus petit flottant strictement positif hors valeurs spéciales. Déterminer leur valeur approximative et la valeur exacte de leur produit sous forme d'une expression simplifiée.

Question P4 : Écrire une fonction de signature `sup_b : bool array -> bool array -> bool` déterminant si le premier tableau de booléens représente un flottant supérieur ou égal au flottant représenté par le deuxième tableau de booléens. Si une valeur spéciale est rencontrée au niveau de l'un des deux arguments, le comportement est au choix.

Question P5 : Écrire une fonction de signature `exposant : float -> int` déterminant le plus grand entier n tel que deux à la puissance n soit inférieur ou égal à l'argument supposé positif (inutile de le vérifier et comportement au choix sinon). Calculer rigoureusement la complexité de cette fonction si une boucle a été utilisée, et l'estimer si une récursion a été utilisée. **Tout calcul explicite de logarithme ou assimilé est interdit ici. Il faut mettre en œuvre un algorithme**

Question P6 : Écrire une fonction de signature `intbin : int -> bool array` convertissant un entier en la séquence de onze bits par la représentation de l'exposant évoquée ci-avant. Le comportement est au choix si l'entier n'est pas dans l'intervalle autorisé.

Question P7 : Écrire une fonction de signature `mantisse : float -> bool array * bool` convertissant un flottant dont on suppose qu'il est compris entre zéro et un en le tableau des cinquante-deux bits après la virgule de l'écriture en binaire du nombre, la valeur de retour s'accompagnant d'un booléen précisant s'il faut procéder à une retenue par la suite.

Question P8 : Écrire finalement une fonction de signature `representation : float -> bool array`, utilisant les fonctions précédentes, encodant un flottant en tableau de booléens selon la norme IEEE754. On se posera bien la question de l'intérêt de la retenue dans l'exercice précédent.

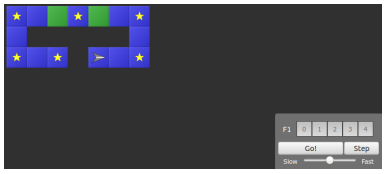
Problème 2 : Robozzle

Une instance de Robozzle est un plateau de jeu contenant des cases de trois couleurs différentes : des bleues, des rouges et des vertes, certaines de ces cases étant marquées d'une étoile, d'un robot présent sur une des cases avec une orientation quelconque parmi les quatre directions cardinales, et d'une liste de fonctions à remplir avec pour chaque fonction une limite d'instructions. Les instructions possibles sont de la forme `si condition alors action`, avec 4 conditions possibles : vrai, la case du robot est bleue, rouge, ou verte. Les actions possibles sont l'avancée d'une case, un virage à gauche sur place, un virage à droite sur place et un appel à n'importe laquelle des fonctions. Une instance est résolue si on parvient à renseigner les fonctions disponibles (pas nécessairement toutes) dans la limite de la place disponible de sorte que le robot passe par toutes les cases marquées d'une étoile (ce qui interrompt immédiatement l'exécution des instructions) sans jamais tomber du plateau.

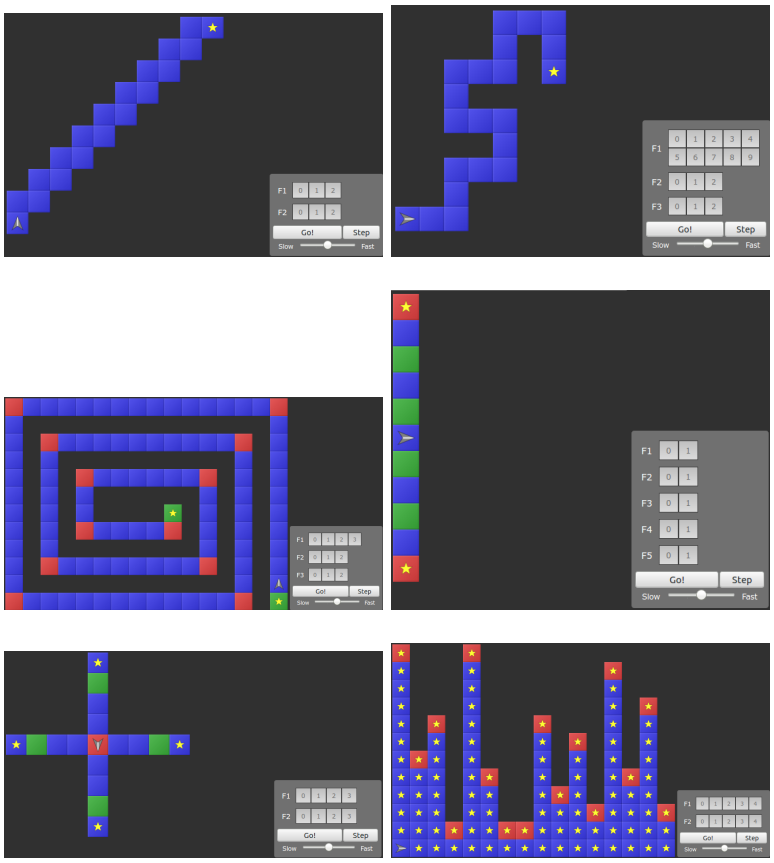
Il est important de noter que le premier appel se fait sur la fonction F1.

À titre d'exemple, l'instance suivante peut se résoudre en fournissant, avec une fonction de 5 instructions disponible :

F1 : si vrai alors avancer (qu'on écrira simplement « avancer »), avancer, tourner à gauche, si la case du robot est verte alors tourner à droite (qu'on écrira simplement « droite vert ») et rappeler F1.



Résoudre les six autres instances du jeu ci-dessous.



Résoudre la dernière montre une belle capacité d'anticipation sur la notion de pile et le chapitre sur la récursivité...