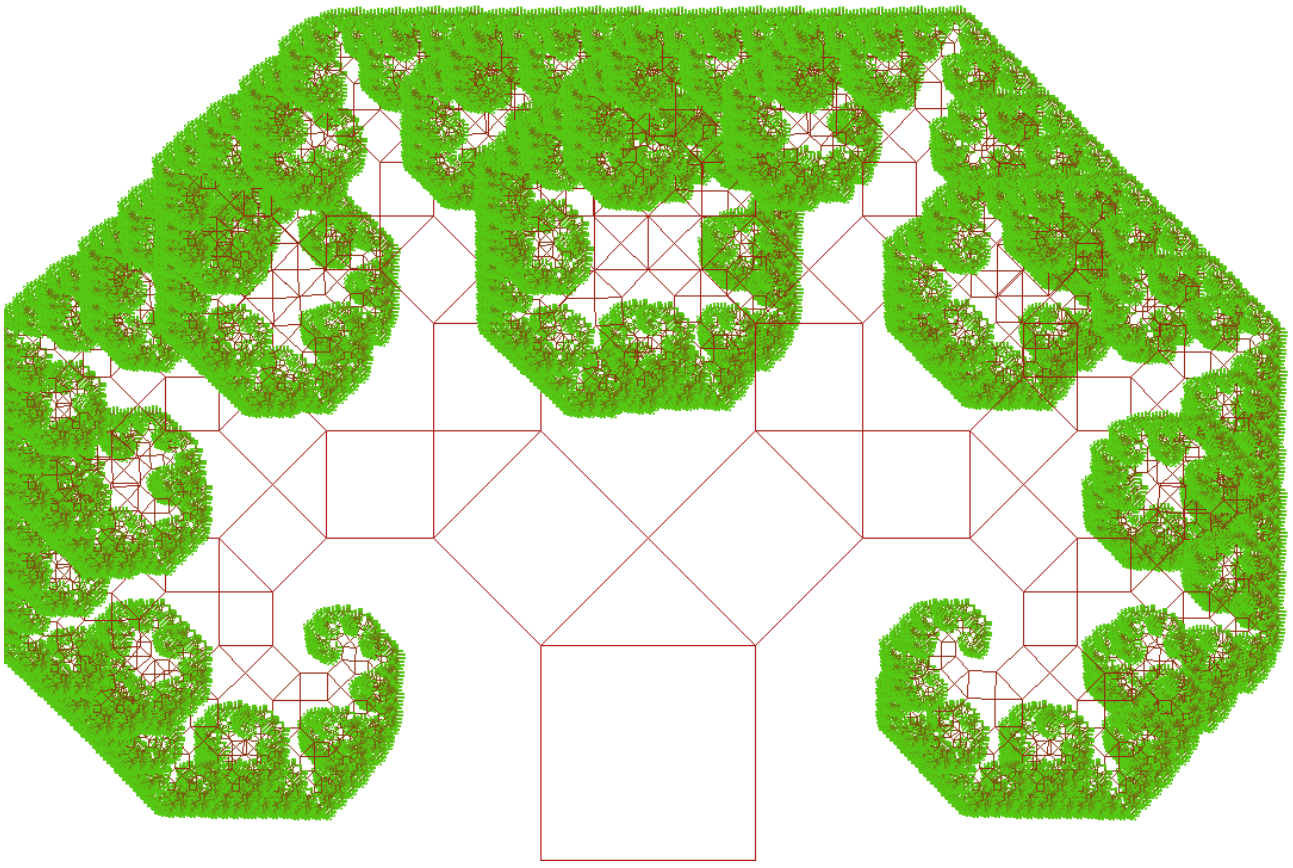
Devoir Non Surveillé 2

À rendre le 8 janvier 2024

INFORMATIQUE MP2I



Tournez la page S.V.P.

1 Étude des nombres flottants – (20 pts)

Dans cette section, on admet (sans le démontrer) que les flottants sont le sous ensemble $\mathbb{F}$ de $\mathbb{R}$ des $n2^e$ avec n et e entiers, $|n| < 2^{53}$ et $-1074 < e < 970$. La spécification des flottants garantit que l'addition sur $\mathbb{F}$, qu'on note $\oplus$, se comporte comme si le calcul avait d'abord été fait dans $\mathbb{R}$ de façon exacte puis qu'on avait arrondi le résultat. Il existe plusieurs modes d'arrondi. On considère ici l'arrondi par défaut, qui est l'arrondi au plus proche (en anglais *round to nearest*). Il s'agit de la fonction $\mathcal{N} : \mathbb{R} \to \mathbb{F}$ qui, pour un réel, x, renvoie le nombre flottant le plsu proche de x. En cas de point milieu (si x est à égale distance de deux nombres flottants), $\mathcal{N}(x)$ est celui qui a une valeur de n paire. On a donc la relation

$$\forall x \in \mathbb{F}, \forall y \in \mathbb{F}, \qquad x \oplus y = \mathcal{N}(x + y)$$

On a des relations similaires pour la soustraction $\ominus$ et la multiplication $\otimes$.

☐ 1 – L'ensemble $\mathbb{F}$ munit de la relation d'ordre usuelle $\leq$ sur les réels forme-t-il un ensemble bien fondé ?

☐ 2 –

1. Prouver que $0, 1, 1/8, 2^{52}$ et -2^{52} sont des flottants selon cette caractérisation, mais pas $2^{52} + 1/8$.
2. Donner le plus grand nombre flottant et le plus petit nombre flottant strictement positif.
3. Donner tous les nombres flottants dans l'intervalle $[1 - 2^{-52}; 1 + 2^{-52}]$.

☐ 3 – Calculer $(-2^{52} \oplus 2^{52}) \oplus 1/8$. Calculer $-2^{52} \oplus (2^{52} \oplus 1/8)$. Conclure sur une propriété non respectée par $\oplus$.

☐ 4 – Trouver un nombre flottant x non nul tel que $x \otimes x = 0$. Que dire du code suivant :

```
if (x != 0) {z = 1/(x*x);}
```

?

☐ 5 –

1. Trouver un nombre flottant x tel que $1 \oplus x = 1$.
2. Trouver le plus grand nombre flottant x tel que $1 \oplus x = 1$. Justifier.

☐ 6 – Si l'on soustrait deux valeurs proches, mais résultats d'un calcul arrondi, on peut obtenir un résultat complètement faux. Comparer $(2^{52} \oplus 1/8) \ominus (2^{52} \ominus 1/8)$ et la valeur mathématique sans arrondi $(2^{52} + 1/8) - (2^{52} - 1/8)$.

☐ 7 – Mais ce n'est pas la dernière soustraction qui créé cette erreur, elle ne fait que mettre en lumière les erreurs précédentes. Prouver le lemme suivant : soient x et y deux nombres flottants

$$\text{si } y/2 \leq x \leq 2y, \text{ alors } x \ominus y = x - y.$$

2 Permutations – C (30 pts)

On représente en C une permutation σ d'un ensemble de n éléments par un tableau de taille n, contenant les entiers de 0 à $n-1$.

☐ 8 – Écrire une fonction `int *compose(int sigma1[], int sigma2[], int n)` qui prend en entrée 2 permutations σ_1 et σ_2 et qui crée et renvoie un tableau d'entiers correspondant à la permutation $\sigma_1 \circ \sigma_2$.

☐ 9 – Écrire une fonction `bool est_involution(int sigma[], int n)` qui prend en entrée une permutations σ et qui renvoie `true` si σ est une involution et `false` sinon.

☐ 10 – Écrire une fonction `bool est_permutation(int sigma[], int n)` qui prend en entrée un tableau d'entiers σ et qui renvoie `true` si σ est bien une permutation et `false` sinon.

On s'intéresse maintenant aux cycles dans une permutation. Par exemple, la permutation `{ 0, 2, 1, 4, 5, 3}` possède 3 cycles : (0), $(1,2)$, et $(3,4,5)$, de longueurs 1, 2, et 3.

☐ 11 – Écrire une fonction `int taille_cycle(int i, int sigma[], int n)` qui prend en entrée un entier i et une permutation σ et qui renvoie la taille du cycle de σ contenant i.

☐ 12 – Écrire une fonction `int nb_cycles(int sigma[], int n)` qui prend en entrée une permutation σ et qui renvoie la nombre de cycles qu'elle contient.

On propose de parcourir toutes les permutations par ordre lexicographique pour conserver celle de poids minimal. Par exemple, la permutation qui suit $(0,2,4,3,1)$ dans l'ordre lexicographique est $(0,3,1,2,4)$.
Si $p = (p_0, p_1, \dots, p_{n-1})$ est une permutation de $[\![0, n-1]\!]$, on définit les indices suivants :
- $j = \max\{i \in [\![0, n-2]\!] \mid p_i < p_{i+1}\}$ et $j = -1$ si cet ensemble est vide.
- $k = \max\{i \in [\![j+1, n-1]\!] \mid p_j < p_i\}$ et $k = n$ si $j = -1$.

☐ 13 – Écrire une fonction `bool permut_suivante(int *p, int n)` qui renvoie `false` si p est la dernière permutation de n éléments dans l'ordre lexicographique, et qui renvoie `true` et modifie p pour qu'il prenne la valeur de la permutation suivante dans l'ordre lexicographique sinon.

☐ 14 – Écrire une fonction `void anagrammes(char mot[])` qui affiche tous les anagrammes d'un mot. On pourra utiliser l'instruction `printf("%s\n", str)` qui affiche la chaîne de caractères `str` suivie d'un retour à la ligne.

☐ 15 – Écrire une fonction `int experience(int n)` qui prend en entrée un entier n et qui compte le nombre de permutations sur n éléments qui n'ont pas de cycle de taille supérieure à $\frac{n}{2}$.

☐ 16 – On considère l'énigme suivante :

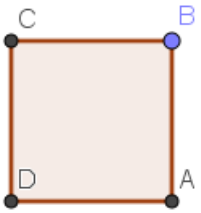
Cent prisonniers (sur une péniche), identifiés par un (unique) nombre entre 1 et 100, ont été condamnés à mort par le cruel Docteur No, qui leur offre cependant une dernière chance. Dans la timonerie, se trouvent 100 tiroirs (numérotés de 1 à 100), où il a aléatoirement mis les 100 numéros des prisonniers. Après concertation, ces derniers ont le droit d'entrer l'un après l'autre et de regarder le contenu de 50 tiroirs (qu'ils referment après leur passage). Ils auront, collectivement, la vie sauve si chacun d'entre eux trouve son numéro dans un tiroir. Malheureusement, ils ne peuvent pas communiquer entre eux et si l'un d'eux échoue, ils seront tous exécutés.
Un mathématicien, pessimiste, se trouve parmi les prisonniers. Selon lui, la probabilité de succès est de $2^{-100} \approx 8 \cdot 10^{-31}$.

Proposer une meilleure solution que la solution naïve proposée par le mathématicien et estimer les chances de survie des prisonniers, en fonction de leur nombre n (ils ouvrent $\frac{n}{2}$ tiroirs).

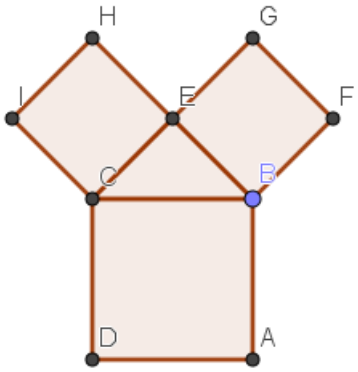
3 Arbres – OCaml (20 pts)

On souhaite dessiner des arbres en OCaml à l'aide du module Graphics. Pour cela, on s'inspire de la construction fractale de l'arbre de Pythagore :

En partant d'un carré ABCD, on lui ajoute un triangle rectangle isocèle BEC dont [BC] est l'hypoténuse, puis on ajoute deux carrés BFGE et EHIC de côtés respectivement [BC] et [EC]. On réitère ensuite cette construction avec les deux carrés obtenus.



Ordre 0



Ordre 1

☐ 17 – Écrire une fonction `carre (xa, ya : int * int) (xb, yb : int * int) : (int * int) * (int * int)` qui à partir des coordonnées des points A et B, calcule les coordonnées des points C et D, trace le carré ABCD, et renvoie le couple de couples $((x_C, y_C), (x_D, y_D))$.

☐ 18 – Écrire une fonction `triangle (xb, yb : int * int) (xc, yc : int * int) : (int * int)` qui à partir des coordonnées des points B et C, calcule les coordonnées du point E, trace le triangle BEC, et renvoie le couple (x_E, y_E).

Nous sommes maintenant prêt à mettre ces fonctions bout-à-bout pour tracer l'arbre de Pythagore. Il nous faut un cas de base pour les appels récursifs, nous allons donc choisir une profondeur d (e.g., 10 au début) jusque laquelle aller, qui décroit à chaque appel récursif, et nous arrêter si $d = 0$.

☐ 19 – Compléter la fonction `arbre` ci-dessous :

```
let rec arbre (a : int * int) (b : int * int) (d : int) =
  if d <= 0 then () else
    begin
      ...
    end
```

☐ 20 – Donner le nombre d'appels à la fonction `arbre` effectués lors du calcul de `arbre a b d`, en fonction de d.

☐ 21 – Pour briser la symétrie de l'arbre, on décide de modifier légèrement l'emplacement des points aléatoirement, par exemple en déplaçant de quelques pixels vers la droite, vers la gauche, vers le haut, vers le bas chaque point avant de le placer définitivement. On pourra pour cela utiliser la fonction `Random.int : int -> int` qui prend en entrée un entier n et renvoie un nombre aléatoire choisi uniformément entre 0 inclus et n exclus.

Proposer une modification des fonctions précédentes afin de tracer une approximation de l'arbre de Pythagore, mais asymétrique, et envoyer vos plus jolis dessins à votre professeur.

Fin de l'épreuve