Introduction à l'algorithmique **Hiver 2025**
Introduction to algorithms ***Winter 2025***

IFT2125-6001 Évaluation *Evaluation* (10%)

Devoir 2 - *Homework 2*

Consignes générales : À faire en équipe de 1 personne minimum et 2 personnes maximum. Remettre un seul pdf et les fichiers de code python et C++. Indiquez vos noms et matricules sur tous les fichiers (pdf et code).
General instructions: *To be done in teams of 1 person minimum and 2 people maximum. Submit only one pdf and the python and C++ code files. Indicate your names and registration numbers on all files (pdf and code).*

Votre code doit obligatoirement passer les tests fournis avec le devoir, sinon une note de 0 sera attribué aux exercices dont les tests de base ne sont pas réussis. D'autres tests seront rajoutés lors de la correction. Seulement les librairies standards de Python et C++ sont permises, sauf si mentionné autrement. Il va de soit que le code doit être clair, bien indenté et bien commenté.
Your code must pass the tests provided with the assignment, otherwise a grade of 0 will be given to exercises whose basic tests are not passed. Other tests will be added during the grading. Only standard Python and C++ libraries are allowed, unless otherwise stated. It goes without saying that the code must be clear, well indented and well commented.

La date de remise est le vendredi 28 avril à 22h30. La structure de la remise dans Studium doit être la suivante :
The due date is Friday, April 28 th at 10:30 PM. *The structure of the submission in Studium should be as follows:*

```
StudiUM
├── devoir2_NOM1_NOM2.pdf
├── tri_hybride.py
├── foresterie.py
├── MinCostRechargeCalculator.cpp
├── MinCostRechargeCalculator.h
├── distribution.py
└── generation_paysage.py
```

Il ne faut **pas** remettre de zip. Également, il ne faut pas modifier les noms de fichiers ou de fonctions (sous risque de perte de points).
*You should **not** submit a zip file. Also, you should not change the names of files or functions (at the risk of losing points).*

1 Trouvez l'erreur *Find the error* (10 points)

Chacune des "preuves" ou des énoncés suivants comporte au moins une erreur. Identifiez clairement où se trouvent toutes ces erreurs, et expliquez de façons appropriée pourquoi vous dites que ce sont des erreurs (par exemple, dites pourquoi une telle étape n'est pas correcte, donnez ce qui manque ou qui devrait être remplacé dans une définition, donnez des contre-exemples invalidant un énoncé, etc).

Each of the following "proofs" or statements contains at least one error. Clearly identify where all these errors are, and explain appropriately why you say they are errors (e.g., state why a given step is incorrect, give what is missing or should be replaced in a definition, give counterexamples that invalidate a statement, etc.).

Question 1: (2 points)

Soit f et g deux fonctions $\mathbb{N} \to \mathbb{R}^{\geq 0}$ telles que $f(n) > g(n)\ \forall n \geq n_0$. Alors $lim_{n\to\infty} \frac{f(n)}{g(n)} = +\infty$.

Let f and g be 2 functions $\mathbb{N} \to \mathbb{R}^{\geq 0}$ such that $f(n) > g(n)\ \forall n \geq n_0$. Then $lim_{n\to\infty} \frac{f(n)}{g(n)} = +\infty$.

Question 2: (2 points)

Soit f, g et h trois fonctions $\mathbb{N} \to \mathbb{R}^{\geq 0}$ telles que $f(n) \geq g(n)\ \forall n \geq n_0$. Si $f(n) \notin \mathcal{O}(h(n))$ alors $g(n) \notin \mathcal{O}(h(n))$.

On peut simplement prendre un exemple avec $f(n) = \frac{n}{10}$, $g(n) = \lfloor \frac{n}{10} \rfloor$ et $h(n) = \sqrt{n}$. On a que $\frac{n}{10} \notin \mathcal{O}(\sqrt{n})$ alors $\lfloor \frac{n}{10} \rfloor \notin \mathcal{O}(\sqrt{n})$.

Let f, g and h be 3 functions $\mathbb{N} \to \mathbb{R}^{\geq 0}$ such that $f(n) \geq g(n)\ \forall n \geq n_0$. If $f(n) \notin \mathcal{O}(h(n))$ then $g(n) \notin \mathcal{O}(h(n))$.

We can simply take an example with $f(n) = \frac{n}{10}$, $g(n) = \lfloor \frac{n}{10} \rfloor$ and $h(n) = \sqrt{n}$. We have that $\frac{n}{10} \notin \mathcal{O}(\sqrt{n})$ then $\lfloor \frac{n}{10} \rfloor \notin \mathcal{O}(\sqrt{n})$.

Question 3: (2 points)

On souhaite homogéniser la récurrence suivante : *We wish to make the following recurrence homogenous*

$$t_n = 4t_{n-1} - 4t_{n-2} + 4(n+1)4^n$$

$$t_n - 4t_{n-1} + 4t_{n-2} = 4(n+1)4^n \quad (\Delta)$$

Multiplions par 4 des deux côtés : *Multiply by 4 each side :*

$$4t_n - 16t_{n-1} + 16t_{n-2} = 4(n+1)4^{n+1}$$

On remplace n par $n-1$: *We replace n by $n-1$*

$$4t_{n-1} - 16t_{n-2} + 16t_{n-3} = 4(n+1)4^n \quad (\square)$$

On soustrait $\Delta - \square$: *We substract $\Delta - \square$*

$$t_n - 4t_{n-1} + 4t_{n-2} - (4t_{n-1} - 16t_{n-2} + 16t_{n-3}) = 4(n+1)4^n - (4(n+1)4^n)$$

Voilà notre récurrence homogène : *Here's our homogenous recurrence*

$$t_n - 8t_{n-1} + 20t_{n-2} - 16t_{n-3} = 0$$

Question 4: (2 points)

Preuve que $2^n \in \Omega(4^n)$: *Proof that $2^n \in \Omega(4^n)$*

En utilisant le critère de la limite, *Using the limit criterion,*

$$\begin{aligned}
\lim_{n\to\infty} \frac{2^n}{4^n} &= \lim_{n\to\infty} \frac{2^n}{2^{2n}} && (1)\\
&= \lim_{n\to\infty} \frac{\lg(2^n)}{\lg(2^{2n})} && (2)\\
&= \lim_{n\to\infty} \frac{n}{2n} && (3)\\
&= \lim_{n\to\infty} \frac{1}{2} && (4)\\
&= \frac{1}{2} && (5)\\
& && (6)
\end{aligned}$$

Comme $\frac{1}{2} \in \mathbb{R}^{>0}$, le critère de la limite nous permet de conclure que $2^n \in \Theta(4^n)$. Ainsi, par définition de $\Theta(4^n)$, nous avons $2^n \in \Omega(4^n)$. ■

Since $\frac{1}{2} \in \mathbb{R}^{>0}$, we conclude by the limit criterion that $2^n \in \Theta(4^n)$. Hence, by definition of $\Theta(4^n)$, we have $2^n \in \Omega(4^n)$. ■

Question 5: (2 points)

Preuve que $2^n \in \Omega(4^n)$: *Proof that* $2^n \in \Omega(4^n)$

Par définition de Ω, on cherche c et n_0 tels que $2^n \geq c4^n, \forall n \geq n_0$. *By definition of* Ω*, we look for* c *et* n_0 *such that* $2^n \geq c4^n, \forall n \geq n_0$.

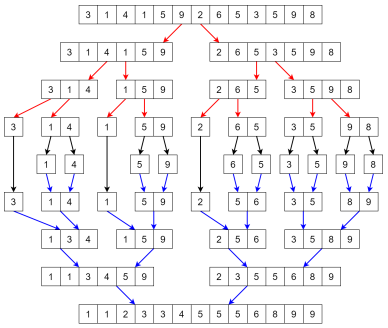
$$2^n \geq c4^n \tag{1}$$
$$\frac{2^n}{4^n} \geq c \tag{2}$$
$$\left(\frac{1}{2}\right)^n \geq c \tag{3}$$
$$\log_{\frac{1}{2}} \left(\frac{1}{2}\right)^n \geq \log_{\frac{1}{2}} c \tag{4}$$
$$n \geq \log_{\frac{1}{2}} c \tag{5}$$

Comme $\log_{\frac{1}{2}} c$ est une constante, en prenant $c = \frac{1}{2}$, on aura $n \geq \log_{\frac{1}{2}} \frac{1}{2} = 1$, ce qui est vrai en prenant $n_0 = 1$. *Since* $\log_{\frac{1}{2}} c$ *is a constant, by taking* $c = \frac{1}{2}$*, we will havea* $n \geq \log_{\frac{1}{2}} \frac{1}{2} = 1$*, which is true with* $n_0 = 1$.

Ainsi, nous avons trouvé $c = \frac{1}{2}$ et $n_0 = 1$ tels que $2^n \geq c4^n, \forall n \geq n_0$. Nous avons donc $2^n \in \Omega(4^n)$. ■

Therefore, we have found $c = \frac{1}{2}$ *et* $n_0 = 1$ *such as* $2^n \geq c4^n, \forall n \geq n_0$. *Hence, we have* $2^n \in \Omega(4^n)$. ■

2 Diviser pour régner - Code Medium *Divide and Conquer - Code Medium* (20 points)



Question: Vous devez implémenter en python3 un hybride de l'algorithme de tri par fusion et tri par insertion pour des nombres. Cet algorithme hybride utilise le concept de diviser pour régner (tri fusion) sauf lorsque la taille du tableau d'entrée est plus petite ou égale à un certain seuil, pour lequel l'algorithme va basculer vers un algo naïf simple (tri par insertion). Dans cet exercice expérimental, votre travail principal est de trouver le seuil optimal (basé sur la taille de l'entrée) pour votre ordinateur. C'est à dire le seuil qui permet d'avoir l'algorithme hybride le plus rapide. Vous devez fournir les informations suivantes dans le devoir (pdf) : OS, mémoire et CPU. Par exemple: Windows 11 Entreprise, 8GB RAM et Intel i5-1135G7. Vous devez aussi fournir une courte description de votre approche, les résultats et une analyse de ces résultats qui vont justifier le seuil que vous avez choisi. Il est recommandé de fournir un tableau qui illustre les vitesses moyennes pour chaque taille d'entrée et pour chaque variante d'algorithme (variante = seuil différent). L'utilisation de ressources automatiques est permise, mais il faut cependant ajouter en annexe la discussion (requêtes et réponses).

You need to implement in python3 a hybrid of merge sort and insertion sort algorithm for numbers. This hybrid algorithm uses the concept of divide and conquer (merge sort) except when the size of the input array is smaller than or equal to a certain threshold, for which the algorithm will switch to a simple naive algorithm (insertion sort). In this experimental exercise, your main task is to find the optimal threshold (based on the size of the input) for your computer. That is, the threshold that allows you to have the fastest hybrid algorithm. You must provide the following information in the assignment (pdf): OS, memory and CPU. For example: Windows 11 Enterprise, 8GB RAM and Intel i5-1135G7. You must also provide a short description of your approach, the results and an analysis of these results that will justify the threshold you have chosen. It is recommended to provide a table that illustrates the average speeds for each input size and for each algorithm variant (variant = different threshold). The use of automatic resources is allowed, but you must add the discussion (queries and answers) in the appendix.

Remise : Compléter le fichier `tri_hybride.py` fournis, et ne remettre **uniquement** que ce fichier. Ne **pas** remettre le fichier `tri_hybride_test.py`, ou n'importe quel autre

fichier de test.

Submission: *Complete the file* `tri_hybride.py` *provided, and submit* ***only*** *this file. Do not submit the file* `tri_hybride_test.py`*, or any other test file.*

3 Programmation dynamique - Code Facile *Dynamic programming - Code Easy* (10 points)

Question:

Grâce à une nouvelle loi environnementale qui réduit la déforestation, une compagnie forestière ne peut plus couper deux arbres qui sont adjacents. Vous devez aider la compagnie forestière à sélectionner les arbres qui pourront être coupé. Pour simplifier le problème, on suppose que les arbres sont positionné dans une rangée de n arbres. Vous recevez un tableau "forest" qui donne la valeur de chaque arbre lorsque coupé. Vous recevez aussi *cost* qui représente le coût constant d'opération à la compagnie pour couper chaque arbre.

Thanks to a new environmental law that reduces deforestation, a logging company can no longer cut down two adjacent trees. You must help the logging company select which trees to cut down. To simplify the problem, assume the trees are positioned in a row of n trees. You are given a table "forest" that gives the value of each tree when cut down. You are also given cost, which represents the constant operating cost to the company for cutting down each tree.

Par exemple, si nous avons le coût d'opération $cost = 2$ et le tableau "forest" suivant :

$[3, 5, 11, 9, 4]$, la solution optimale est de prendre les arbres avec valeurs 3, 11 et 4 qui après avoir enlevé les coût d'opérations auront les valeurs 1, 9 et 2 pour un total de 12 profit. Résolvez le problème en Python 3.
For example, if we have the operation cost cost $= 2$ *and the following "forest" array:* $[3, 5, 11, 9, 4]$*, the optimal solution is to take the trees with values 3, 11 and 4 which after removing the operation costs will have the values 1, 9 and 2 for a total of 12 profit. Solve the problem in Python 3. Solve the problem in Python 3.*

Complexité recherchée : $\mathcal{O}(n)$ où n est la taille d'entrée. Votre algorithme devrait être efficace pour avoir tous les points.
Desired complexity: $\mathcal{O}(n)$ *where* n *is the input size. Your algorithm should be efficient to get all points.*

Code
Une partie du code est déjà écrite. Dans le fichier `foresterie.py`, il faut compléter la fonction `solve()`. Vous pouvez écrire des fonctions auxiliaires au besoins. Un fichier `foresterie_test.py` et des fichiers d'entrées vous sont fournis pour vous aider à tester votre code.
Some of the code is already written. In the `foresterie.py` *file, you need to complete the* `solve()` *function. You can write auxiliary functions if needed. A* `foresterie_test.py` *file and input files are provided to help you test your code.*

Exemple d'appel (dans le dossier où se trouvent `foresterie.py` et `foret1.txt`):
Example call (in the folder where `foresterie.py` *and* `foret1.txt` *are located):*

```
python3 foresterie.py foret.txt
```

Remise : Compléter le fichier `foresterie.py` fournis, et ne remettre **uniquement** que ce fichier. Ne **pas** remettre le fichier `foresterie_test.py`, ou n'importe quel autre fichier de test.
Submission: *Complete the file* `foresterie.py` *provided, and submit* ***only*** *this file. Do not submit the file* `foresterie_test.py`*, or any other test file.*

4 Programmation dynamique - Code Facile *Dynamic programming - Code Easy* (10 points)

Question:

Un camion électrique doit se rendre d'une ville à une autre. Sur le chemin, des bornes de recharge électrique sont installées à distances égales une de l'autre. Cependant, les prix d'accès à la borne de recharge sont variables. Une fois à la borne, la quantité d'électricité que le camion peut prendre est illimitée. La capacité limitée de la batterie restraint le camion à se déplacer à un maximum de 3 bornes. Le camion part complètement chargé. Les distances entre le point de départ et la première borne, puis entre dernière borne et le point d'arrivée sont toutes égales à la distance entre 2 bornes consécutives. Les prix des accès à chaque borne est indiqué dans un tableau $RechargeCost$ de taille n. Quelle est le coût minimal pour faire le trajet? Résolvez le problème en C++.

An electric truck needs to travel from one city to another. Along the way, electric charging stations are installed at equal distances from each other. However, the prices for accessing the charging station vary. Once at the station, the amount of electricity the truck can take is unlimited. The limited battery capacity restricts the truck to traveling to a maximum of 3 stations. The truck leaves fully charged. The distances between the starting point and the first station, then between the last station and the arrival point are all equal to the distance between 2 consecutive stations. The prices for accessing each station are indicated in an array RechargeCost of size n. What is the minimum cost to complete the trip? Solve the problem in C++

Complexité recherchée : $\mathcal{O}(n)$ où n est le nombre de bornes de recharge. Votre algorithme

devrait être efficace pour avoir tous les points.
Desired complexity: $\mathcal{O}(n)$ where n is the number of charging stations. Your algorithm should be efficient to have all the points.

Code
Exemple d'appel :
Example call:

```
Compilation :
g++ -o .\min_cost_recharge.exe .\min_cost_recharge.cpp .\MinCostRechargeCalculator.cpp

Execution :
.\min_cost_recharge.exe
```

Remise : Compléter les fichiers `MinCostRechargeCalculator.cpp` et `MinCostRechargeCalculator.h` fournis, et ne remettre **uniquement** que ces fichiers. Ne **pas** remettre le fichier `min_cost_recharge.cpp`.
Submission: *Complete the provided* `MinCostRechargeCalculator.cpp` *and* `MinCostRechargeCalculator.h` *files, and only submit* ***these*** *files. Do not submit the* `min_cost_recharge.cpp` *file.*

5 Arbres - Code Facile *Trees - Code Easy* (10 points)

Question:

Vous recevez un arbre binaire en entrée avec n noeuds. Chaque noeud de l'arbre possède entre 0 et n jetons. Le nombre total de jetons dans l'arbre est de n. Une transaction se définie comme un déplacement d'un jeton d'un noeud à un autre noeud adjacent (de enfant à parent ou de parent à enfant). Combien faut-il de transactions pour arriver à un arbre binaire où chaque noeud possède un seul jeton? Résolvez le problème en Python 3.

You are given a binary tree as input of n nodes. Each node ot the tree contains between 0 and n tokens. The total number of tokens in the tree is n. A transaction is defined as a movement of a token from one node to another adjacent node (from child to parent or from parent to child). How many transactions does it take to arrive at a binary tree where each node has a single token? Solve the problem in Python 3.

Complexité recherchée : $\mathcal{O}(n)$ où n est la taille d'entrée. Votre algorithme devrait être efficace pour avoir tous les points.

Desired complexity: $\mathcal{O}(n)$ where n is the input size. Your algorithm should be efficient to get all points.

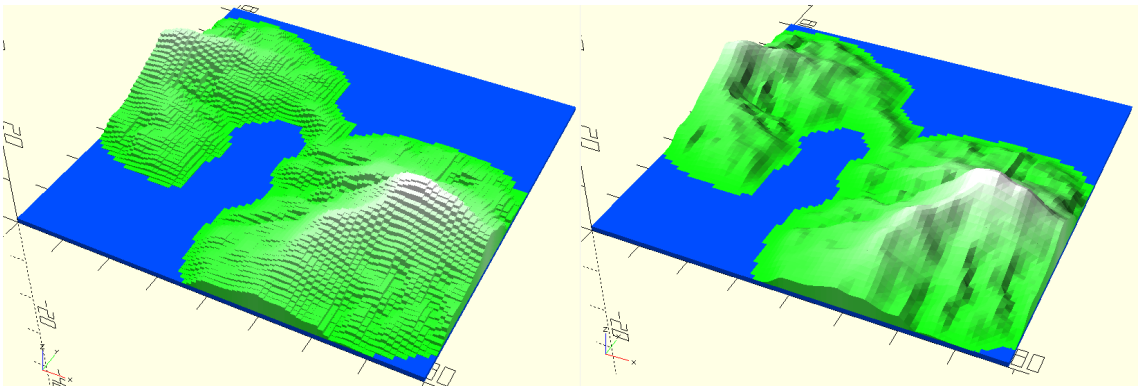
Code
Une partie du code est déjà écrite. Dans le fichier `distribution.py`, il faut compléter la fonction `solve()`. Vous pouvez écrire des fonctions auxiliaires au besoins. Un fichier `distribution_test.py` et des fichiers d'entrées vous sont fournis pour vous aider à tester votre code. L'entrée dans les fichiers de test (par exemple `tree1.txt`) est sous forme de liste mais est transformée en arbre. La conversion se fait en prenant les éléments de la liste un par un dans l'ordre et en les plaçant dans l'arbre de haut en bas et de gauche à droite. La structure de donnée de l'arbre est `TreeNode`, définie dans le fichier `distribution.py`. Pour vous aider, il y a une fonction (rudimentaire) de visualisation nomée `print_tree()`.
Some of the code is already written. In the `distribution.py` *file, you need to complete the* `solve()` *function. You can write auxiliary functions if needed. A* `distribution_test.py` *file and input files are provided to help you test your code. The input in the test files (for example* `tree1.txt`*) is a list but is transformed into a tree. The conversion works by taking each element of the list and placing them one by one in the tree, from top to bottom, left to right. The tree data structure is* `TreeNode`*, defined in the file* `distribution.py`*. To help you, there is a (rudimentary) visualisation function called* `print_tree()`*.*

Exemple d'appel (dans le dossier où se trouvent `distribution.py` et `tree1.txt`):
Example call (in the folder where `distribution.py` *and* `tree1.txt` *are located):*

```
python3 distribution.py tree1.txt
```

Remise : Compléter le fichier `distribution.py` fournis, et ne remettre **uniquement** que ce fichier. Ne **pas** remettre le fichier `distribution_test.py`, ou n'importe quel autre fichier de test.
Submission: *Complete the file* `distribution.py` *provided, and submit* ***only*** *this file. Do not submit the file* `distribution_test.py`*, or any other test file.*

6 Génération de paysage en 3D - Code Difficile *3D landscape generation- Code Difficult* (40 points)



Vous devez générer un paysage 3D d'une ou plusieurs îles dans l'océan avec OpenSCAD en écrivant un programme en python. Il y a des fonctionnalité obligatoires à implémenter et des fonctionalités optionnelles. Pour avoir tous les points il faut implémenter toutes les fonctionalités obligatoires et au moins 4 parmis les fonctionalités optionnelles. Dans le rapport pdf du devoir 2, vous devez expliquer brièvement votre approche et chacun des algorithmes utilisés. Des fichiers exemple sont fourni avec le devoir. *You must generate a 3D landscape of one or more islands in the ocean in OpenSCAD by writing a Python program. There are mandatory features to implement and optional features for which you must implement at least 4 to get full points. In the PDF report for assignment 2, you must briefly explain your approach and each of the algorithms used. Example files are provided with the assignment.*

L'utilisation de ressources automatiques est permise, mais il faut cependant ajouter en annexe la discussion (requêtes et réponses). Bonus de +10% au devoir si un paysage 3D généré par votre code est imprimé et présenté à un membre de l'équipe enseignante (le membre devra prendre votre oeuvre en photo). L'impression peut se faire gratuitement à l'atelier de fabrication numérique de la bibliothèque math-info. Une mini formation sera offerte à la classe. *The use of automated resources is permitted, but the discussion (requests and answers) must be appended. A 10% bonus to the assignment is provided if a 3D landscape generated by your code is printed and presented to a member of the teaching team (the member must take a photo of your work). Printing can be done free of charge at the math-info library's digital fabrication workshop. A mini-training session will be offered to the class.*

Fonctionnalité obligatoires : *Mandatory features:*

1. Fichier model.scad généré dans le même dossier que le code python *Generated model.scad file in the same folder as the python code*
2. Océan (plaque de base) et île(s) *Ocean (base plate) and island(s)*

3. Pente générale qui va des sommets des îles à la côte *General slope from the island peaks to the coast*
4. Matrice des hauteurs de taille entre 30×30 et 100×100 *Height matrix of size between 30×30 and 100×100*
5. Résultat dans OpenSCAD de taille entre 50×50mm et 80×80mm, hauteur maximale de 40mm *Result in OpenSCAD between 50×50mm and 80×80mm, maximum height 40mm*
6. Style cubique (comme le jeu Minecraft) ou maillage polyédrique *Cubic style (like Minecraft) or polyhedral mesh*
7. Initiales des concepteurs + "IFT2125" en dessous de la plaque de base *Designers' initials + "IFT2125" below the base plate*

Minimum 4 parmi les fonctionnalité optionnelles suivantes : *Minimum of 4 optional features from the following:*

1. Ajout de bruit aléatoire dans l'élévation *Adding random noise to the elevation*
2. Utilisation des couleurs dans OpenSCAD *Using colors in OpenSCAD*
3. Île qui n'est pas un cercle parfait ou élipse *Island that is not a perfect circle or elipse*
4. Rivières ou vallées *Rivers or valleys*
5. Lacs intérieurs *Inland lakes*
6. Falaises et plages sur différents endroits de la côte *Cliffs and beaches at different locations along the coast*
7. Volcan *Volcano*
8. Grotte ou arche (attention lors de l'impression 3D) *Cave or arch (be careful with 3D printing)*
9. Végétation (arbres, forêt, buissons) *Vegetation (trees, forest, bushes)*
10. Structures humaines (bâtiments, routes, port, observatoire astronomique) *Human structures (buildings, roads, harbor, astronomical observatory)*
11. Baie ou lagune *Bay or lagoon*
12. Élévation non linéaire par rapport à la côte *Nonlinear elevation relative to the coast*
13. Tuiles hexagonales à la place de carrées *Hexagonal tiles instead of square ones*

Pour valider une fonctionnalité, l'aspect de réalisme, de proportions et d'effort sera pris en compte. Par exemple, ajouter un simple et unique prisme rectangulaire fixé au même endroit, ne pourra valider la fonctionalité "structure humaine". *To validate a feature, the aspect of realism, proportions and effort will be taken into account. For example, adding a simple and unique rectangular prism fixed in the same place will not be able to validate the "human structure" feature.*

Bonus impression 3D à la bibliothèque math-info : *3D printing bonus at the math-CS library*

- Lire la documentation avant de poser des questions aux bibliothécaires *Read the documentation before asking questions to the librarians*
- Pour du soutien avec les imprimantes 3D demandez aux bibliothécaires *For support with the 3D printers, ask the librarians*
- Max 3 heures pour l'impression de votre paysage 3D *Max 3 hours to print your 3D landscape*
- Épaisseur de couche entre 0.2mm et 0.3mm *Layer thickness between 0.2mm and 0.3mm*
- Remplissage entre 5% et 20% *Infill between 5% and 20%*
- Restez voir la 1ère couche imprimée *Stay to see the 1st printed layer*
- Récuppérez votre paysage une fois l'impression terminée *Take your landscape once the printing is done*
- Remplissez le registre papier des impressions *Fill in the paper registar of printings*

Remise : Créer et compléter le fichier `generation_paysage.py`, et ne remettre **uniquement** que ce fichier.
Submission: *Create and complete the file* `generation_paysage.py`*, and submit* ***only*** *this file.*

Voici un exemple de procédure pour générer un paysage. Vous n'êtes pas tenu de suivre cet ordre. Vous pouvez laisser aller votre créativité! *Here's an example of how to generate a landscape. You don't have to follow this order. You can let your creativity run wild!*

1. Distrinction entre terre et mer *Distinction between land and sea*
2. Calcul de l'élévation *Calculation of the elevation*
3. Ajout de bruit aléatoire dans l'élévation *Addition of random noise to the elevation*
4. Lissage *Smoothing*

5. Ajout de rivières ou vallées *Adding rivers or valleys*

6. Résultat final après 2e lissage - version cubique *Final result after 2nd smoothing - cubic version*

7. Résultat final après 2e lissage - version maillage polyhedral *Final result after 2nd smoothing - polyhedral mesh version*