Devoir en temps limité n°1 – 2h (calculatrice autorisée)

On veillera à présenter très clairement sa copie : il faut rédiger les réponse et encadrer les résultats. Pour le code, il doit être indenté, on ne commence pas une fonction en bas de page et on utilise de la couleur pour les commentaires.

Il est attendu que vous expliquiez votre code dès lors qu'il est conceptuellement compliqué ou qu'il prend plus de 5 lignes. Sauf mention contraire, il est possible d'utiliser les fonctions écrites dans les fonctions des questions suivantes.

Exercice 1 Questions de cours

1. Comment sont représentées les chaînes de caractères en C ?
2. Considérer l'instruction C suivante : `double x = 3/2;`. Quelle est la valeur de la variable `x` ?
3. Qu'est-ce qu'un pointeur ? Quelle bout de code C permet de désigner "la valeur pointée par le pointeur p" ?
4. Créer un tableau statique de taille 5 qui contient 0,1,2,3,4 .
5. Qu'est-ce qu'un variant ? À quoi ça sert ? Donner un variant de la boucle dans le code suivant, avec une justification :

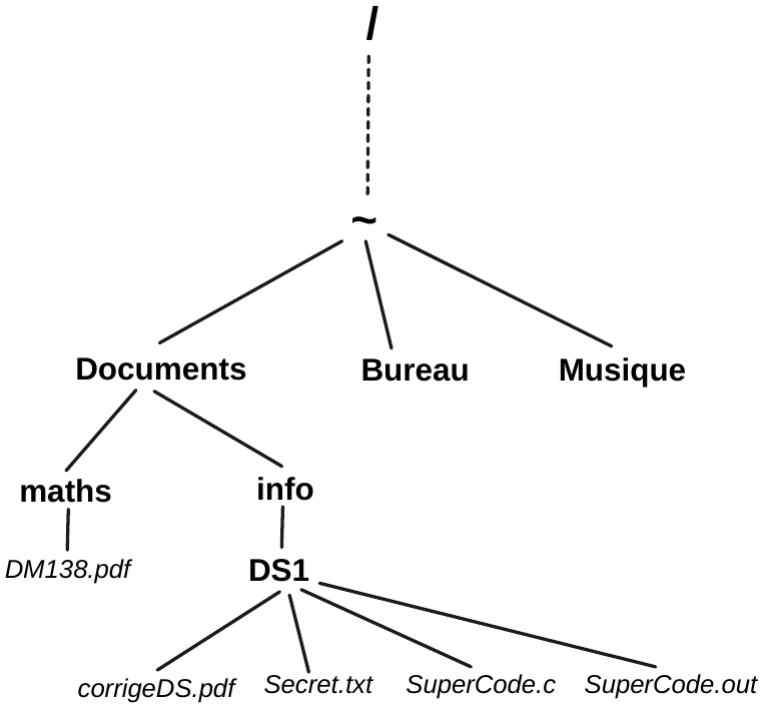
```c
int x = 0;
int y = 3;
while (x<34){
   x+=1;
   y-=1;
}
```

Exercice 2 Binaire

6. Traduire le nombre suivant en décimal, on fera apparaitre la méthode : $\overline{10100101}^2$.
7. Traduire les nombres suivants en binaire, seul le résultat est attendu : 10, 125, 245.
8. Qu'est-ce que le complément à 1 d'un nombre binaire ?
9. Rappeler la méthode pour faire une soustraction.
10. Faire les calculs suivants sur un octet : $\overline{10100101}^2 + \overline{111101}^2$, $\overline{10100111}^2 - \overline{00111101}^2$.

Exercice 3 Bash

Imaginons que sur votre ordinateur, vous avez l'arborescence de fichiers suivantes. Les noms en gras représentent des dossiers et les noms en italique représentent des fichiers.



11. Expliquer ce qu'est le dossier / et ce qu'on y trouve.

On démarre le terminal, qui se place à ∼.

12. (a) Écrire une ou des instructions bash pour se déplacer jusqu'au dossier **maths**
    (b) Écrire une instruction affichant le contenu du dossier.
    (c) Écrire une instruction créant un nouveau fichier **theoreme.txt**
    (d) Écrire une instruction pour se déplacer une fois en arrière dans l'arborescence.
    (e) Écrire une instruction créant un nouveau dossier **SI**.
13. (a) Écrire une ou des instructions pour retourner à ∼.
    (b) Écrire une ou des instructions pour se déplacer dans le dossier **DS1**
    (c) Écrire une instruction permettant d'afficher le contenu du dossier, y compris les permissions.

On suppose que la commande précédente donne le résultat suivant dans le terminal :

```
-rw-rw-r-- 1 moi MP2I 18444 10 janv. 2020 corrigeDS.pdf
-rw-rw---- 1 moi MP2I 10001 30 déc. 23:23 Secret.txt
-rw-r--r-- 1 moi MP2I 1234 12 mai 2023 SuperCode.c
-rw-r--r-- 1 moi MP2I 256 12 mai 2023 SuperCode.out
```

14. Expliquer les permissions du fichier **corrigeDS.pdf** . Qui a quelles permissions ?
15. Modifier les permissions du fichier **Secret.txt** pour qu'il ne puisse être lu, modifié et executé que par vous (l'utilisateur).
16. Écrire une instruction pour donner à tout le monde la permission d'exécuter **SuperCode.out** .

Exercice 4 Du C

17. Dessiner l'état de la mémoire associé au code suivant. On demande un dessin à chacun des endroits où se trouve un trait

```c
int n = 12;
int* p = &n;
int* q = malloc(sizeof(int));
*q = 5;
*p += (*q)*2;
----------------------------------------------------------------(Dessin ici)
int m = *q;
q = p;
m += 3;
*q = *p+1;
----------------------------------------------------------------(Dessin ici)
int** pp = &p;
**pp = 5;
if(*q == 4){*q+=1;}
q=NULL;
----------------------------------------------------------------(Dessin ici)
```

18. Écrire une fonction `int somme(int n)` qui calcule et renvoie la somme $\Sigma_{i=0}^{n-1}(4 * i^3 + 2 * i^2)$.
    Dans cette question il est interdit d'utiliser une formule du cours de maths pour simplifier la somme.
19. Écrire une fonction `int* tableau_pair(int n)` qui crée et renvoie un tableau dynamique de taille n contenant, dans l'ordre, $0, 2, 4, 6, 8, \ldots, 2n-2$.
20. Écrire une fonction `bool est_dans(double x, double* t, int n)` qui renvoie **true** si x est présent dans le tableau t et **false** sinon. n est la taille du tableau t.
21. Écrire une fonction `int nb_occurences(double x, double* t, int n)` qui renvoie le nombre de fois où x est présent dans le tableau t.
22. Écrire une fonction `double le_plus_courant(double* t, int n)` qui renvoie l'élément le plus courant (celui qui apparait le plus de fois) du tableau t. En cas d'égalité on peut renvoyer un élément quelconque parmi ceux qui ont le plus d'apparitions.

Exercice 5 À propos de notes

On considère une classe avec n étudiants qui effectuent un devoir (qu'on appelera le DS1). Les étudiants portent des numéros entre 0 et $n-1$ (pour les anonymiser).

On range les notes des étudiants (fictifs) dans un tableau `double* tnotes`, créé (et rempli) dans le `main`. La case i du tableau contient la note de l'étudiant i.

23. Écrire une fonction `double moyenne(double* t, int n)` permettant de calculer la moyenne des valeurs d'un tableau t de taille n.
    Que faut-il écrire dans le `main` pour afficher la moyenne des notes de nos étudiants fictifs ?
24. Écrire une fonction `int major(double* t, int n)` permettant de savoir quel étudiant a obtenu la meilleure note au devoir. On renverra le numéro de l'étudiant.
25. Écrire une fonction `int rang(double* t, int n, int numetu)` prenant en entrée le tableau des notes, sa taille et un numéro d'étudiant et renvoyant le rang de l'étudiant dans la classe.
    Précision : supposons que 3 étudiants aient la note 10 et que 12 étudiants aient strictement plus que 10. Alors les 3 étudiants auront le même rang 13. Le ou les étudiants ayant la première note strictement inférieure à 10 auront le rang 16.

On dit qu'une note est unique si elle n'a été obtenue que par un seul étudiant.

26. Écrire une fonction `void uniques(double* t, int n)` qui affiche (dans n'importe quel ordre) les notes uniques des étudiants.

Voici la définition de l'écart-type σ d'un ensemble de notes $\{x_0, x_1, ..., x_{n-1}\}$: $\sigma = \sqrt{\frac{1}{n}\Sigma_{i=0}^{n-1}(x_i - \bar{x})^2}$, où $\bar{x}$ est la moyenne des notes.

En C, on peut utiliser la fonction `double sqrt(double x)` pour calculer une racine carrée. On peut utiliser `double pow(double valeur, double puissance)` pour calculer une puissance. On supposera que la bonne librairie est incluse.

27. Écrire une fonction `double ecart-type(double* t, int n)` qui calcule l'écart-type du tableau de notes.

On définit également la médiane d'un ensemble de notes $\{x_0, x_1, ..., x_{n-1}\}$. C'est une valeur telle que 50% des notes lui sont inférieures (au sens large) et 50% des valeurs lui sont supérieures.

- Dans le cas où $n = 2k + 1$ est impair, il s'agit de trouver la $k + 1$-ème valeur dans l'ordre croissant.
- Dans le cas où $n = 2k$ est pair, on prend la moyenne de la k-ème valeur et la $k + 1$-ème.

Par exemple la médiane de $\{13, 13, 10, 11, 12, 14, 13, 14, 15\}$ est 13, car 10,11,12,13 lui sont inférieures et 13,14,14,15 lui sont supérieures (au sens large). Remarque : ici 13 est présent en 3 exemplaires, la 5ème valeur dans l'ordre croissant est un 13 et les autres 13 sont placés respectivement dans le groupe des éléments plus petits et le groupe des éléments plus grands.

La médiane de $\{5, 12, 7, 15, 16, 10, 9, 9, 12, 11\}$ est $10,5$ car il y a 10 valeurs et la 5ème est 10 et la 6ème est 11. La médiane est donc $(10 + 11)/2 = 10,5$.

28. Écrire une fonction `double mediane(double* t, int n)` qui calcule la médiane du tableau de notes.
    Remarque : vous êtes libres de la méthode, avoir une bonne complexité n'est aucunement un but.