TRAVAUX PRATIQUES III

Salade composée

Ce TP contient moults exercices de programmation. Le but est de consolider vos acquis. Les entrées du programme seront souvent données au format suivant : un ou plusieurs entiers qui définissent la taille du problème, puis des entrées dont le nombre dépend de ces premiers entiers. Par exemple, la première ligne peut contenir un entier qui indique combien d'entrées sont présentes sur la ligne suivante :

Terminal

```
3
52 47 82
```

Dans cet exemple, la première ligne contient 3 : il y aura donc 3 entrées sur la ligne suivante, et c'est bien le cas.

1. Comment faire une fonction qui lit un entier $0 \leq n \leq 1000$ puis n autres valeurs ?

Il pourra être judicieux pour gagner du temps de ne pas créer un nouveau `.c` à chaque question, mais plutôt de créer une nouvelle fonction résolvant la question et de l'appeler depuis `main` .

La section A est pensée pour les débutants en code, les programmeurs expérimentés peuvent commencer directement par la B. Vous pouvez ne pas traiter les questions d'une section dans l'ordre.

Un·e MP2I averti·e en vaut deux

GCC peut détecter des erreurs potentielles et en avertir l'utilisateur. On utilise à cette fin les deux otpions `-Wall` (*ALL Warnings*) et `-Wextra` (*EXTRA Warnings*) . Ces avertissements sont importants ! Ils indiquent fréquemment des erreurs, parfois un défaut de lisibilité du code. Un bon programme compile sans avertissements [1].

À partir de maintenant, tous vos programmes devront être compilés avec les options `-Wall` et `-Wextra` !

A Salade

2. Écrire une fonction `int max_3(int a, int b, int c)` qui renvoie le maximum de trois entiers.

3. Écrire une fonction `bool pythagore(int a, int b, int c)` qui teste si trois entiers strictement positifs vérifient $\texttt{a}^2 + \texttt{b}^2 = \texttt{c}^2$.

4. **a.** Écrire une fonction `void ligne_etoile(int l)` qui affiche `l` caractères '*' consécutifs.

b. Écrire une fonction `void carre_etoile(int l)` qui affiche un carré de dimensions `l` $\times$ `l` de caractères `'*'`

5. Écrire une fonction qui lit un entier $0 \leq n \leq 1000$ puis n entiers `int` , et renvoie le plus petit entier impair parmis ces entiers.

6. Écrire une fonction `void alternance(int l)` qui affiche en alternance `l` lignes contenant 42 fois le caractère `'+'` et `l` contenant 32 fois le caractère `'-'`

7. Écrire une fonction qui prend en argument un entier $0 \leq n \leq 1000$, puis n entiers $h_0, \dots, h_{n-1}$ tels que $\forall i, -1000 \leq h_i \leq 1000$ et enfin un entier $-1000 \leq s \leq 1000$ et renvoie le nombre de h_i tels que $h_i \leq s$

8. On définit la suite de Syracuse [2] d'un entier $N \in \mathbb{N}^*$ comme :

$$u_0 = N\,;\quad u_{n+1} = \begin{cases} 3 \times u_n + 1 \text{ si } u_n \text{ est impair,} \\ \frac{u_n}{2} \text{ sinon} \end{cases}$$

On suppose [3] que pour tout $N \in \mathbb{N}^*$, la suite de Syracuse de N contient 1 .

1. Ne me forcez pas à rajouter `-Werror` qui transformerait tous vos warnings en errors…
2. Syracuse comme la ville américaine, pas la ville sicilienne. Le mathématicien fondateur de l'étude de cette suite est Lothar Collatz.
3. Cette conjecture s'appelle la conjecture de Collatz.

a. Si $u_k = 1$, calculez u_{k+1}, u_{k+2}, u_{k+3} . Commentez.

b. Écrire une fonction `uint64_t syracuse_temps_de_vol(int N)` qui prend en argument un entier `N` et renvoie le plus petit indice i tel que $u_i = 1$.

B Crudités

D'ici à la fin du TP, sauf indication contraire, les entiers donnés en entrée sont compris entre -32767 et +32767 .

Dans cette section et les suivantes, il est utile de commencer par se demander : « *Mais que diable doit faire mon programme ? Quelle propriété précise doit-il vérifier, ou quelle action précise doit-il effectuer ? Comment formaliser ce qui est attendu en termes mathématiques ?* »[4] .

9. Écrire une fonction qui lit un entier $0 < n \leq 1000$ puis n entiers $x_0, \dots, x_{n-1}$ de type `int` , et renvoie $\min_{0 \leq i,j \leq n-1} |x_i - x_j|$

10. Une suite (finie ou infinie) $(u_n)_{0 \leq n \leq M}$ est dite p-périodique, $p \in \mathbb{N}^*$, si : $\forall n, 0 \leq n \leq M - d \implies u_n = u_{n+p}$.

Écrire une fonction qui lit un entier $0 < n \leq 1000$ puis n entiers compris entre -32767 et +32767 et enfin un entier p du même intervalle, et teste si les n entiers forment une suite p-périodique.

11. Écrire une fonction qui teste si un tableau de taille $1 \leq \ell < 1000$ est une permutation, c'est à dire s'il contient une et une seule fois chaque entier de $[\![1; \ell]\!]$.

12. Écrire une fonction qui lit deux entiers $0 < m, n \leq 1000$ puis m entiers $x_0, \dots, x_{m-1}$ puis n entiers $y_0, \dots, y_{n-1}$, et qui teste si $\{x_0, \dots, x_{m-1}\} \subseteq \{y_0, \dots, y_{n-1}\}$

13. Pour $(m, n) \in \mathbb{N}^2$, on définit :

$$u(m,n) = \begin{cases} 2^n \text{ si } m = 0 \\ m \times u(m-1, n^2) \text{ sinon} \end{cases}$$

Écrire une fonction qui lit deux entiers $0 < m, n \leq 255$ et renvoie $u(m, n)$. On ne fera pas une fonction récursive[5].

C Sauce

Les exercices de cette section sont techniques à rédiger et/ou requièrent de bonnes idées algorithmiques.

14. Écrire une fonction qui lit un entier $0 < n \leq 1000$ puis n entiers, et qui en une seule boucle calcule le maximum et second maximum de ces entiers.

15. On dit qu'une suite finie de 0 et de 1 est équilibrée si elle contient autant de 0 que de 1. Un facteur d'une suite finie (u_k) est une sous-suite finie de (u_k) de termes consécutifs.

Écrire une fonction qui lit un entier $0 < n \leq 1000$ puis n entiers valant soit 0 soit 1 et qui renvoie la longueur du plus long facteur équilibré.

16. Écrire une fonction qui lit un entier $0 < n \leq 1000$ puis $2 * n$ entiers : $d_0, f_0, \dots, d_{n-1}, f_{n-1}$, tels que $\forall i, d_i \leq f_i$. Ces entiers représentent n intervalles entiers fermés : $[\![d_0, f_0]\!], \dots$ etc. La fonction devra renvoyer un entier maximal pour le nombre d'intervalles intersectés.

17. Écrire une fonction qui lit un entier $0 < n \leq 1000$ puis n entiers et affiche ces entiers dans l'ordre croissant.

18. Écrire une fonction qui lit un entier n, puis affiche tous les sous-ensembles de $[\![0; n]\!]$. Il s'agit avant tout de trouver comment représenter et énumérer ces sous-ensembles.

D Aromates

Ces exercices sont *très* difficiles. Idéaux pour occuper vos dimanches après-midi pluvieux.

19. Résoudre ce problème du projet Euler[6] : `https://projet-euler.fr/p/135` .

20. **a.** Écrire un programme qui résoud SOMME-PARTIELLE, c'est à dire qui lit un entier $0 < n < 2^{64}$ puis n entiers $x_0, \dots, x_{n-1}$ et ensuite un entier supplémentaire t. Le programme doit afficher s'il existe $I \subseteq [\![0; n-1]\!]$ tel que $\sum_{i \in I}^{x_i} = t$; c'est à dire s'il est possible d'atteindre exactement t en sommant des entiers de $x_0, \dots, x_{n-1}$.

4. Ne jugez pas trop durement mes questionnement existentiels : ce sont maintenant également les vôtres.
5. Si vous ne comprenez pas cette consigne, tout va bien. Si par contre vous êtes frustrés par cette interdiction : cf note de bas de page 5
6. Le projet Euler réunit moults exercices d'informatique, proches des mathématiques. Attention ils sont souvent difficiles !

b. Rendre votre programme précédent efficace [7].

c. Écrire un programme qui résoud #SOMME-PARTIELLE, c'est à dire qui compte le nombre de façons de résoudre SOMME-PARTIELLE.

21. Prouvez la conjecture de Collatz, c'est à dire l'hypothèse de la question 8. [8].

7. Si vous y parvenez, vous avez résolu le troisième problème du millénaire (pour une certaine définition de «efficace»).
8. Ne vous épuisez pas trop non plus, c'est un problème ouvert de mathématiques.