TP 6 : Fonctions (partie 2)

Consignes

— Le TP est à faire de manière individuelle.
— Il est demandé de rendre une archive `nom_prenom.zip` qui contient les fichier `.c` et `.py`.
— L'archive doit être rendue sur Moodle à la fin des 4h de TP.
— Il est conseillé d'utiliser un éditeur directement sur le PC (e.g. Visual Studio Code).

Exercice 1 : Jeu d'Allumettes

Chaque joueur prend à tour de rôle de 1, 2 ou 3 allumettes dans un tas comprenant initialement x allumettes. Celui qui prend la dernière allumette perd le jeu.

- **Premier Cas** :

Les deux joueurs sont des êtres humains. La fonction qui gère le jeu prendra en paramètre le nombre initial d'allumettes ainsi que le numéro du joueur qui commence. Lors de chaque tours, le nombre d'allumettes à prendre par le joueur est demandée par une saisie sur le clavier.

1. Implémenter le programme en langage C. L'utilisation des fonction rendra votre programme de meilleure qualité. Le fichier s'appellera `allumette1.c`.
2. Proposer une implémentation en langage Python. Les fichiers s'appelleront `allumette1.py`.

Sur la console, l'exécution du programme peut vous donner, par exemple, la figure ci-dessous :

```
Le nombre restant d'allumettes dans le jeu est egal a 15
Quel est le nombre d'allumettes prises par le joueur 2 ?
2
Le nombre restant d'allumettes dans le jeu est egal a 13
Quel est le nombre d'allumettes prises par le joueur 1 ?
3
Le nombre restant d'allumettes dans le jeu est egal a 10
Quel est le nombre d'allumettes prises par le joueur 2 ?
4
Votre choix n'est pas valide.
Quel est le nombre d'allumettes prises par le joueur 2 ?
3
Le nombre restant d'allumettes dans le jeu est egal a 7
Quel est le nombre d'allumettes prises par le joueur 1 ?
2
Le nombre restant d'allumettes dans le jeu est egal a 5
Quel est le nombre d'allumettes prises par le joueur 2 ?
1
Le nombre restant d'allumettes dans le jeu est egal a 4
Quel est le nombre d'allumettes prises par le joueur 1 ?
3
Bravo !!! Le joueur 1 a gagne.
Appuyez sur une touche pour continuer...
```

- **Second Cas** :

L'un des deux joueurs peut être l'ordinateur. Vous devez alors programmer son jeu. Pour simplifier, l'ordinateur tirera au hasard le nombre d'allumettes à prendre.

3. Implémenter le programme en langage C. Le fichier s'appellera `allumette2.c`.
4. Implémenter le programme en langage Python. Le fichier s'appellera `allumette2.py`.

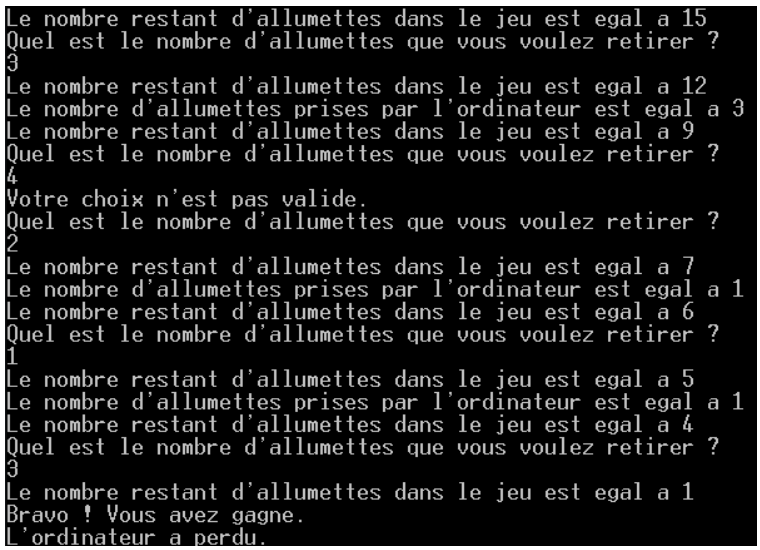
Indices :

Pour générer un nombre aléatoire sous C, il y a la fonction `rand()` qui est présente dans la bibliothèque `stdlib.h`.

Cependant, pour obtenir quelque chose de vraiment aléatoire, il faut rajouter dans votre programme la ligne d'instruction suivante : `srand(time(NULL));`. Ne pas oublier d'inclure la bibliothèque `time.h`.

Par exemple, pour retourner un nombre aléatoire entier compris entre 10 et 20 inclus, il faut saisir l'instruction suivante : `nbaleatoire=10+rand()%11;`.

Sur la console, l'exécution du programme peut vous donner, par exemple, la figure ci-dessous :



Exercice 2 : Création d'une bibliothèque personnalisée (finir l'exercice précédent en priorité)

Voici une méthode **en C** permettant de créer une bibliothèque personnelle par exemple `libmat.a` contenant des fonctions.

Cette méthode traitera des bibliothèques statiques, qui seront directement mises dans l'exécutable une fois la compilation effectuée.

Ces bibliothèques ont une extension `.a` et se trouvent généralement dans le répértoire `TDM-GCC-64/lib/`.
Soit un fichier `fonction.c` contenant des fonctions.
Pour utiliser ces bibliothèques, il faut faire les étapes suivantes :

1. Compiler chaque source avec l'instruction : `gcc -c fonction.c`
2. Créer, par exemple, la bibliothèque `libmat.a` avec l'instruction : `ar r libmat.a fonction.o`
3. Générer l'index avec l'instruction `ranlib libmat.a`
4. Écrire les en-têtes des fonctions dans le fichier `libmat.h` comme ceci :
   `extern int toto(int a,float b);`
5. Placer la bibliothèque (`libmat.a`) dans le sous-répertoire `lib/` et l'en-tête (`libmat.h`) dans le sous répertoire `include/`.
6. Pour compiler un fichier source appelant cette bibliothèque, il faut demander à l'éditeur de liens d'utiliser `libmat.a` à l'aide de l'argument `-lmat`.
   Par exemple, `gcc toto.c -lmat`

Pour créer votre propre module **en Python** (bibliothèque de fonctions), il suffit de programmer les fonctions qui le constituent dans un fichier portant le nom du module, suivi du suffixe `.py`.

Depuis un (autre) programme en Python, il suffit alors d'utiliser la primitive `import` pour pouvoir utiliser ces fonctions.

Pour appeler une fonction ou une variable de ce module, il faut que le fichier `xx.py` soit dans le répertoire courant (dans lequel on travaille) ou bien dans un répertoire listé par la variable d'environnement PYTHONPATH de votre système d'exploitation.

Appliquer ces méthodologies sur l'exercice précédent pour créer des bibliothèques en C et en Python permettant d'utiliser ces fonctions.