Lycée Michel Montaigne,Bordeaux *Octobre 2022*

DS INFO N°1

Partie pratique

Vous créerez en tout premier lieu un dossier `DS1` dans votre répertoire de travail habituel sur votre machine, qui contiendra tout votre travail de la séance.

Tous les fichiers source seront à remettre sur la plateforme Moodle en cliquant sur

`DS 1 - Remise des codes source`

dans le cours intitulé `INFORMATIQUE - Devoirs`.

Remettez toujours un travail, même partiel ou qui n'a pas abouti.

S'il vous faut choisir, privilégiez la qualité du code à la quantité. Préférez rendre deux codes qui fonctionnent plutôt que quatre codes qui ne fonctionnent pas.

Pour l'ensemble des fichiers à remettre, vous devez respecter la convention de nommage des fichiers en remplaçant `NOM` par votre propre nom de famille en majuscule.

Toute navigation sur le Web est interdite.

Seul l'accès au cours `INFORMATIQUE - Devoirs` et aux ressources qui y ont été déposées est autorisé.

Toutes les consultations de documents sont enregistrées, je pourrai donc voir si d'autres ressources que celle de la page Moodle dédié à ce DS ont été consultées ou téléchargées durant le devoir.

Exercice 1 (Programmation en OCaml).

Écrire dans un script OCaml `NOM_ds1.ml` les fonctions suivantes. Pour chaque fonction, vous écrirez dans le script au moins deux tests pour valider votre travail.

1. La fonction mathématique à variable réelle $g(x) = x + 5$; testez au moins une valeur positive et une valeur négative.
2. La fonction mathématique à variable réelle $f(x) = 5x^2 + 3$; testez au moins une valeur positive et une valeur négative.
3. La fonction `g_rond_f`, équivalent de la fonction mathématique $g \circ f$, en utilisation les deux définitions précédentes ; testez au moins une valeur positive et une valeur négative.
4. Une fonction `affiche_parite` qui prend en entrée un entier relatif et qui affiche `Le nombre xxx est pair` s'il est pair et `Le nombre xxx est impair` dans le cas contraire. Dans ces affichages, `xxx` désigne le nombre passé en entrée de la fonction.

Exercice 2 (Programmation en C).

Écrire en C un programme `NOM_binary.c` qui prend en entrée un nombre **entier naturel fourni par l'utilisateur sur la ligne de commande** et **affiche à l'écran le nombre de bits minimal nécessaire pour encoder ce nombre**.
Par exemple, si l'utilisateur du programme tape la ligne de commande :

```
./binary 38
```

le code doit afficher :

```
Il faut au minimum 6 bits pour encoder la valeur 38.
```

Attention, vous n'avez pas le droit d'importer d'autres bibliothèques que `stdio` et `stdlib`.

Par souci de modularité, l'algorithme de calcul du nombre de bits d'encodage sera codé à part dans une fonction `nombre_bits_minimal` qui prendra en entrée l'entier naturel donné par l'utilisateur et retournera en sortie le nombre minimal de bits nécessaires à l'encodage.
Cette fonction sera appelée dans la fonction principale de votre programme. L'affichage écran du résultat se fera dans le programme principal.

Compilez, testez, déboguez votre programme.

Exercice 3 (Techniques de débogage et tests).

Téléchargez le code `DS1_code_a_deboguer.ml` disponible dans le cours Moodle `INFORMATIQUE - Devoirs` en faisant clic-droit puis `Enregistrer la cible du lien sous...`.

Ranger le fichier téléchargé dans votre dossier `DS1`.

Déboguez pas à pas ce code.

Vous remettrez le code débogué sous le `NOM_code_ok.ml` sur Moodle. Remettez un code même s'il n'est pas complètement débogué.

Vous pouvez mettre des commentaires dans le code pour expliquer vos difficultés et déductions si vous en éprouvez la nécessité (non obligatoire !)

Exercice 4 (BONUS - Programmation en C).

Cet exercice doit être traité en dernier si les 3 premiers exercices ont été correctement traités. Il viendra juste bonifier une note déjà bonne.

On appelle miroir d'un nombre entier naturel l'entier obtenu en inversant l'ordre de ses chiffres. Par exemple, le miroir de 1234 est 4321.

Écrire en C un code `NOM_miroir.c` qui renvoie l'entier miroir d'un entier naturel. L'utilisateur pourra fournir en entrée sur la ligne de commande l'entier naturel de son choix. Par exemple, la ligne de commande :

```
./miroir 1133322
```

doit aboutir à l'affichage du message

```
Le nombre miroir de 1133322 est 2233311
```

Par souci de modularité, l'algorithme de calcul du nombre miroir sera codé à part dans une fonction `miroir` qui prendra en entrée l'entier naturel donné par l'utilisateur et retournera en sortie le nombre miroir. Cette fonction sera appelée dans la fonction principale de votre programme. L'affichage écran du résultat se fera dans le programme principal.

Compilez, testez, déboguez et commentez votre programme.