IN 101 - TD 02 **Énoncé**

Attention : désormais l'implémentation des exercices sera à faire en fin de TD, après que nous ayons examiné toutes les questions de réflexion et de structuration des algorithmes. Laissez le clavier...

1 Division

≪Écrivez une fonction qui calcule et affiche le quotient de 2 entiers pour un nombre de décimales donné, sans 0 non significatifs en fin d'affichage. ≫

Q 1.1. Quels sont les domaines d'entrée et de sortie de cette fonction ?

Q 1.2. Que pensez-vous de l'utilisation de l'opérateur `/` pour le but recherché ?

Q 1.3. Quelles sont les parties qui composent un affichage résultat ?

Q 1.4. Un programme devant réagir correctement à toutes les configurations de données qui lui sont soumises, quelle doit être la première vérification de votre programme ?

Q 1.5. Comment allez-vous calculer la partie entière du quotient ?

Q 1.6. Pour la partie décimale, comment sait-on s'il y en a une à afficher et comment obtient-on les chiffres ?

Q 1.7. Comme l'on souhaite ne pas voir s'afficher de 0 inutiles en queue de résultat, à quelle condition le calcul (et donc de l'affichage) des décimales doit-il continuer ?

Q 1.8. Donnez la forme de l'algorithme de la fonction `division`.

Q 1.9. Que se passe-t-il si le dividende ou le diviseur sont négatifs ? Si besoin, rectifiez l'algorithme.

2 Tri

≪Écrivez une fonction vérifiant si un tableau d'entiers est trié. ≫

Q 2.1. Quels sont les domaines des entrées et des sorties ?

Q 2.2. Que signifie ≪est trié ≫ ? Quel va être le choix qui devra être fait pour l'algorithme ?

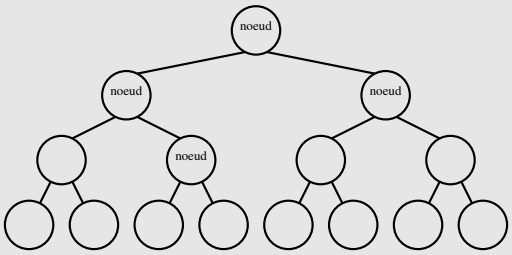
Q 2.3. Définissez formellement le critère ≪être trié ≫ pour un tableau. Quel est l'impact de ≪strict ≫ ou ≪ou égal ≫ ?

Q 2.4. En vous appuyant sur la définition formelle d'un tri vue en question Q2.3, proposez la structure d'un algorithme répondant à la question du tri.

3 Arbre

≪Écrivez une fonction permettant d'afficher un arbre binaire complet de manière textuelle. ≫

En informatique, un arbre est une structure où des *nœuds* sont reliés à d'autres nœuds (*fils*) de manière arborescente (cf. IN103). Un arbre est *binaire* si chaque nœuds **a au plus 2** fils. Un arbre est *complet* si tous ses nœuds **terminaux** ≪sont au même niveau ≫. Un schéma valant mieux qu'un long discours :



On souhaite afficher uniquement les nœuds d'un tel arbre avec des espaces et le caractère `'*'` en fonction d'une hauteur h donnée en argument de la fonction. Les noeuds à la base de l'arbre sont séparés par **1 seul espace**.

Exemples :

— Pour une hauteur 2 :

```
 *

* *
```

— Pour une hauteur 3 :

```
   *

 *   *

* * * *
```

— Pour une hauteur 4 :

```
     *

   *   *

 *   *   *   *

* * * * * * * *
```

Q 3.1. Quels sont les domaines des entrées et des sorties ?

Q 3.2. De quoi est composé l'affichage ? De quelles relations avez-vous besoin pour faire cet affichage ?

Q 3.3. Donnez l'expression formelle de ces relations.

Q 3.4. Donnez la structure grossière de l'algorithme.

4 Implémentation

Écrivez en `C` les algorithmes esquissés dans les exercices 1, 2 et 3. Pour l'exercice 2, un squelette de programme vous est donné (`sorted_skel.c`) comportant de nombreux cas de test (fastidieux à écrire).

S'il vous reste du temps ou pour continuer après la séance.

5 Chaîne d'heure

≪Écrivez une fonction qui **affiche** la chaîne de caractères représentant une heure telle que l'on la dit à l'oral en français. ≫

Le cours débute à ≪quinze heures moins le quart ≫. Le TD commence à ≪seize heures ≫. On change de jour à ≪minuit ≫. À ≪une heure et demi ≫, il fait nuit, pareil à ≪deux heures moins vingt cinq ≫.

On souhaite ainsi obtenir la chaîne de caractères représentant une heures ≪à l'ancienne ≫, en ne disant pas ≪quarante cinq ≫ ou ≪trente ≫. On vous épargne la gestion des tirets dans les nombres composés (≪quarante cinq ≫ au lieu de ≪quarante-cinq ≫ ira très bien).

Le but n'est pas de faire 2 grosses suites de `if/else if`, une pour les 23 heures et une pour les 59 minutes ! Ceci n'aurait aucun intérêt algorithmique et serait fort fastidieux.

La solution de cet exercice n'est pas donnée dans ce PDF car le code source est un peu long. Vous pourrez néanmoins consulter ce dernier puisqu'il figure dans l'archive solution qui vous est fournie.

Q 5.1. Quels sont les cas particuliers des heures ?

Q 5.2. Quels sont les cas particuliers des minutes ?

Q 5.3. Réfléchissez à ce que vous pouvez factoriser dans l'algorithme pour réutiliser le même traitement dans différents cas. Attention, il traîne encore quelques cas particuliers de la langue française non mentionnés en Q5.1.

Q 5.4. Programmez la fonction `string_of_time` demandée. Attention à ne pas rajouter d'espaces inutiles. Vous pouvez tester votre fonction avec (au moins) les cas suivants :

`string_of_time (12, 0)` → "midi"
`string_of_time (00, 0)` → "minuit"
`string_of_time (00, 15)` → "minuit et quart"
`string_of_time (00, 30)` → "minuit et demi"
`string_of_time (00, 42)` → "une heure moins dix huit"
`string_of_time (12, 0)` → "midi"
`string_of_time (12, 45)` → "treize heures moins le quart"
`string_of_time (21, 21)` → "vingt et une heures vingt et une"
`string_of_time (7, 30)` → "sept heures et demi"
`string_of_time (1, 25)` → "une heure vingt cinq"
`string_of_time (12, 20)` → "midi vingt"
`string_of_time (17, 35)` → "dix huit heures moins vingt cinq"
`string_of_time (23, 59)` → "minuit moins une"
`string_of_time (23, 25)` → "vingt trois heures vingt cinq"