Exercices pratiques de révision :
Pointeurs, lecture d'un fichier, processus (*fork*) et communications inter-processus (*pipe*)

1 Manipulation des pointeurs en C

1.1 Un début

Soit la suite des définitions :

```
int a = 10;
int b = 25;
int * p = &b;
int *pp = &a;
```

Donner le résultat de chacune des expressions suivantes après l'exécution de cette suite de définitions. En particulier, il est demandé de fournir un résultat sur papier, et uniquement après de réaliser une implémentation affichant les résultats. L'objectif est de vous permettre de vérifier et comprendre vos premiers résultats.

1. *(&(*(*(&p))))
   - ◇ la valeur de a
   - ◇ la valeur de b
   - ◇ l'adresse de a
   - ◇ l'adresse de b
   - ◇ la valeur de p
   - ◇ la valeur de pp
2. *(p-1)
   - ◇ la valeur de a
   - ◇ la valeur de b
   - ◇ l'adresse de a
   - ◇ l'adresse de b
   - ◇ la valeur de p
   - ◇ la valeur de pp
3. *(*(&p)-1)
   - ◇ la valeur de a
   - ◇ la valeur de b
   - ◇ l'adresse de a
   - ◇ l'adresse de b
   - ◇ la valeur de p
   - ◇ la valeur de pp
4. *(*(&pp)+1)
   - ◇ la valeur de a
   - ◇ la valeur de b
   - ◇ l'adresse de a
   - ◇ l'adresse de b
   - ◇ la valeur de p
   - ◇ la valeur de pp
5. *(&(*(*(&p))) - 1)
   - ◇ la valeur de a
   - ◇ la valeur de b
   - ◇ l'adresse de a
   - ◇ l'adresse de b
   - ◇ la valeur de p
   - ◇ la valeur de pp

1.2 Tabeaux dynamiques

Ecrire une fonction *extract* qui prend en paramètre :
— T un tableau d'entiers
— n la taille du tableau T.
— a et b deux entiers
et qui renvoie un nouveau tableau contenant uniquement les éléments de T compris entre a et b. Exemple : si T = [1 ; 3 ; 5 ; 7 ; 9], a = 2 et b = 5, on doit obtenir le nouveau tableau [3 ; 5]. Dans un programme testant votre fonction, afficher les éléments du tableau obtenu.

1.3 Calcul sur un tableau

Ecrire et tester une fonction récursive qui calcule la somme des éléments d'un tableau. La taille et les éléments du tableau sont à saisir depuis l'entrée standard dans le programme principal (en utilisant par exemple *scanf(..)*).
Remarque : la fonction n'a besoin que de deux paramètres (pas plus).

2 Lecture d'un fichier à contenu structuré

Le but de cet exercice est de parcourir le contenu d'un fichier pour extraire une information donnée. Il est question en particulier de lire un fichier image au format PGM binaire.
Le fichier PGM binaire est constitué d'une entête ASCII suivi des octets de l'image ligne par ligne et de gauche vers la droite. L'entête a le format suivant (toujours en ASCII) :

— 1 nombre magique du format sur 2 octets : dans le cas présent : P5
— Des éventuels commentaires précédés par le caractère #
— 2 entiers indiquant la hauteur et la largeur de l'image (en nombre de pixels)
— 1 entier indiquant la valeur maximale (profondeur) des pixels

Exemple :

P5
commentaires
300 700
255

Dans un fichier imagePGM.c, implémenter les fonctions définies dans le fichier imagePGM.h (fourni). Ensuite, écrire un programme dans un fichier testPGM.c pour tester les fonctions implémentées.

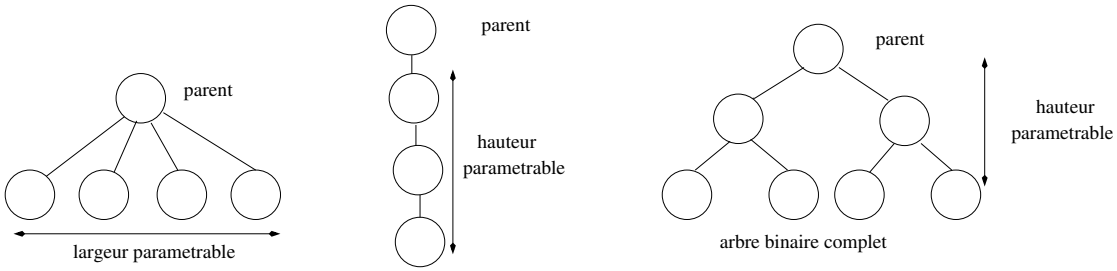
Ecrire un programme permettant de lire une image et de copier l'image dans un nouveau fichier "copie.pgm". S'assurer de la correction de votre programme en comparant une image en entrée avec sa copie générée. Pour cette comparaison, utiliser la commande $diff$.

Voici un ensemble de fonctions recommandées pour la manipulation des fichiers : $fopen$(...), $fread$(...), $fscanf$(...), $fgetc$(...), $fseek$(...), $fprintf$(...), $fwrite$(...) et $fclose$(...)

Remarque : l'utilisation de $fscanf$ suppose un bonne maîtrise du comportement de la fonction $scanf$

3 Processus : fork()

On souhaite mettre en place des applications composées de processus suivant différents schémas. Pour chaque schéma représenté :



1. Réaliser l'application correspondante. Les données largeur et hauteur doivent être des paramètres de votre programme. Pour la simulation de longs calculs, les processus devront s'endormir quelques secondes/minutes en utilisant l'appel système sleep().
2. Vérifier que le schéma est bien respecté. Pour cela, afficher l'arborescence de votre application à l'aide d'un terminal (commande pstree -p).
3. Observer et vérifier les valeurs des variables héritées par chaque processus.

4 Diffusion avec tubes

On veut mettre en place un système de diffusion de messages dans une application composée de plusieurs processus suivant le schéma d'arbre binaire complet.

En utilisant les tubes, modifier le programme développé dans la section 3 pour permettre à un message (saisi au clavier), d'être diffusé dans toute l'arborescence (en partant de la racine).