Examen final - LO01 P2011

2 H - Sans documents (sauf diagrammes de C) ni calculette

N'oubliez pas de commenter vos programmes.
Les types des paramètres et des fonctions sont parfois volontairement omis dans l'énoncé.
A vous de les définir.

Vous aurez besoin de 3 copies : une par exercice

1. Transformation d'Images (7 points) → copie n°1

Définissons une image comme étant une structure contenant les dimensions de l'image (nombre de lignes et nombre de colonnes) et un tableau 2Dimentions de pixels. Le nombre maximum de lignes et de colonnes est 128. Un pixel est représenté par 3 valeurs pour chacun des canaux RVB (Rouge Vert Bleu) et un coefficient de transparence. Les valeurs de R, V, B sont des entiers compris entre 0 et 255.
Le coefficient de transparence est un réel compris entre 0 et 1.

On voudrait écrire quelques fonctions permettant de manipuler ces images.

Définitions :
Pour transformer une couleur en niveau de gris on utilise la formule suivante :

G= 0,299R+ 0,587V+0,114B
R=V=B=G

Pour mélanger 2 couleurs, on applique la formule suivante

C3= min(C1 + C2*a1, W)
a3 = a1*a2

C1, C2, C3 sont des triplets RVB et W représente la couleur blanche R= 255, V=255 B=255. La formule s'applique alors à chacune des composantes RVB. C1 représente la couleur RVB de l'image 1, a1 son coefficient de transparence. C2 représente la couleur RVB de l'image 2, a2 son coefficient de transparence. C3 est la couleur résultante et a3 est le coefficient de transparence résultant du mélange.

Soit le type *unPixel* défini de la manière suivante :
```
typedef struct {
    int R,V,B;      /* composantes Rouge Vert Bleue */
    float a;        /* coefficient de transparence */
    }unPixel;
```

1) Définir le type *uneImageCoul* (définition au début de l'énoncé).

2) Ecrire la fonction *grayScale(…ImG, …ImC)* qui calcule une image en niveau de gris *ImG* à partir de l'image couleur *ImC*.

3) Ecrire la fonction *rotate90(…ImC)* qui applique à l'image couleur passée en paramètre une rotation de 90° dans le sens trigonométrique.

4) Ecrire la fonction *blendCol(…ImC1, …ImC2, …ImC3)* qui calcule l'image *imC3* résultant du mélange de 2 images couleurs *ImC1* et *ImC2* données en paramètres. Si les tailles ne sont pas identiques, le mélange se fera sur la plus petite partie commune (qui correspondra à la taille de l'image résultante).

5) Supposons connues les fonctions suivantes :
   *void initImage(uneImageCoul *ImC)* qui initialise une image couleur,
   void afficheImage (uneImageCoul ImC) qui affiche une image couleur.
   Ecrire le programme qui initialise 2 images couleurs et affiche l'image résultant de leur mélange, sur laquelle on appliquera ensuite une rotation de 90° avant de la transformer en niveau de gris. On affichera ensuite ce résultat final.

Rappel : T[i][j] est la composante d'un tableau T à 2 dimensions en ligne i et colonne j.

2. Agence immobilière (7 points) → copie n°2

Une agence immobilière voudrait stocker dans un fichier l'ensemble de ses biens. Chaque bien étant décrit par :

- Une adresse
- Un type (nombre de pièces)
- Un tableau avec la superficie de chaque pièce
  (le nombre maximum de pièces est de 20)

Une adresse est décrite par :

- Un numéro
- Un nom de rue
- Un nom de ville
-

Exemple de bien : 12 square chanel Compiègne
3
25 12 45

25 étant la superficie de la pièce 1, 12 la superficie de la pièce 2 et 45 la superficie de la pièce 3.

1) Définir le type *unBien* basé sur une structure, correspondant à un bien. On définira au préalable le type *uneAdresse*.
2) Ecrire la fontion *initUnBien()* qui initialise un bien à partir de données entrées par un utilisateur.
3) Ecrire la fonction *initUneAgence()* qui aura comme paramètre un nom de fichier. Cette fonction crée et remplit le fichier de biens à vendre. Ce ficher est associé à une agence dont le nom est passé en paramètre.
4) Ecrire la fonction calculeSuperficie() qui calcule la superficie d'un bien passé en paramètre.
5) Ecrire le programme principal qui initialise le fichier "agence-du-parc.dat", demande à un client une superficie et affiche le nombre de biens correspondants.

3. Exercices et question de cours (6 points) → copie n°3

3.1 Récursivité : La multiplication égyptienne

La multiplication égyptienne de deux entiers naturels m et n est basée sur l'addition et la division par 2. Le principe est le suivant :

- m * n est égal à 0 si n vaut 0
- m * n est égal à m + m * (n – 1) si n est impair
- m * n est égal à (m + m) * (n / 2) si n est pair

Ecrire une fonction **récursive** permettant de réaliser la multiplication égyptienne de 2 nombres entiers.

3.2 Chaînes et tableaux : brouillage

Soit la fonction *permute(char *a, char *b)* qui permute 2 caractères. On veut « brouiller » une chaîne de caractères en effectuant des permutations dans la chaîne de la façon suivante : on permute les 2 premiers caractères puis les 2 suivants et ainsi de suite. Si le nombre de caractères est impair le dernier caractère est inchangé.
Ecrire la fonction *brouilleChaine()* qui brouille une chaîne passée en paramètre selon la méthode décrite ci-dessus en utilisant la fonction *permute()*

Exemple : brouilleChaine de **"Vivement les vacances"** donne **"iVevemtnl sev canaecs"**

3.3 Question de cours

Comment fait-on en langage C pour définir un tableau dont le nombre de composantes est connu en cours d'exécution ? Décrire en quelques lignes.