جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL

كلية العلوم والتقنيات
FACULTÉ DES SCIENCES ET TECHNIQUES

Parcours : MIP (S5)
Examen : Algorithmique et Programmation II
Date : 17/01/2018
Heure : 16H30-18H30
Durée: 2H
Responsables : F.AMOUNAS & B.OUCHAO

Exercice 1.

A) Ecrire en langage C les fonctions suivantes :

1) « **Delete**» ayant une chaîne et un caractère en paramètres et qui supprime tous les caractères identiques au caractère passé en paramètre. Si le caractère spécifié ne fait pas partie de la chaîne, celle-ci devra être retournée intacte.
Par exemple :
Delete ("Bonjour",'o') renverra "Bnjur".
Delete ("examen alg", 'y') renverra " examen alg".

2) « **Start** » et « **End** » qui prennent une chaîne CH et un entier n en paramètres et qui renvoient une chaîne contenant respectivement les n premiers caractères et les n derniers caractères de CH.

3) « **Middle** » qui prend une chaîne et deux entiers p et q en paramètres et qui renvoie la partie de la chaîne comprise entre les caractères p et q.

B) Définir une fonction en langage C, recevant comme paramètre un vecteur V de n éléments, et qui place les éléments de V par ordre croissant dans une matrice M de (r×s) de manière spiralée. L'ordre spiralé revient à parcourir la matrice depuis l'élément (0,0) de manière concentrique dans le sens horlogique.

Exemple : n=9, V <3, 11, 23, 5, 18, 1, 21, 7, 15>, r=3, s=3 et M après le traitement contient :

$$M=\begin{pmatrix} 1 & 3 & 5 \\ 21 & 23 & 7 \\ 18 & 15 & 11 \end{pmatrix}$$

Exercice 2.

Une image est représentée par une matrice de pixels, valeur entière positive sur 8 bits (unsigned char).

a) Ecrire la fonction « *allouer_image* » qui alloue dynamiquement une image.

b) L'histogramme est un tableau d'entiers t dont chaque élément t[i] est le nombre de fois qu'apparaît la valeur i dans l'image. La dimension du tableau d'entiers t est le nombre de valeurs possibles de l'image. Un pixel prend des valeurs comprises entre 0 et 255.
Écrire la fonction « *histo_image* » qui retourne l'histogramme et réalise les opérations suivantes :

- Allocation dynamique de l'histogramme
- Calcul de l'histogramme de l'image im.

c) Eroder consiste à parcourir toute l'image et à trouver le minimum du voisinage du pixel (les 8 pixels voisins) contient au moins un point qui n'est pas un trou. On remplace le pixel par ce minimum.

$$E(x,y)=\min (I(x+dx, y+dy)) \quad dx,dy \in [-1,1]$$

Ecrire la fonction « *erode_image* » qui alloue dynamiquement une image, calcule l'erosion de l'image et retourne cette image. On ne traite pas les pixels au bord de l'image.

d) Le patch P(x,y) de dimension n, situé en x, y est la partie de l'image dont les coordonnées sont [x-n,x+n]. La distance entre le patch P(x,y) et le patch P(u,v) est l'écart quadratique moyen :

$$d^2(P(x,y),P(u,v))=1/k*\sum\sum(P(x+dx,y+dy)-P(u+dx,v+dy))^2$$

ou k est le nombre de pixels de l'image.

Ecrire une fonction *« distance »* qui retourne la distance entre deux patchs de taille n.
NB : il faut exclure les points qui seraient en dehors de l'image.

Exercice 3.

L'algorithme de Lempel-Ziv est un algorithme de compression de données basé sur une généralisation du principe suivant :
On dispose d'un dictionnaire dans lequel tous les mots sont indexés (de 0 à N-1). Il est initialement vide et les mots rencontrés sont ajoutés au fur et à mesure. Chaque fois que l'on reconnaît un mot dans le texte on le remplace par son indice dans le dictionnaire.

Exemple :

Pour compresser le texte « **aabrabdbabrabdg.... »**

a) on lit ce texte de gauche à droite et on le découpe en une suite de mots distincts :

a **ab** **r** **abd** **b** **abr** **abdg**...

Le premier mot est "a", le second "ab" puisque "a" est connu, puis " r", puis "abd" puisque "ab"existe déjà, … on indexe les mots dans l'ordre d'apparition.

b) Chacun des mots se décompose en un mot déjà vu suivi d'une lettre, par convention le mot vide est déjà vu et possède l'indice 0. Le texte précédent est transformé en :

(0,a) (1,b) (0,r) (2,d) (0,b) (2,r) (4,g) ...
a **ab** **r** **abd** **b** **abr** **abdg**...
1 2 3 4 5 6 7

Mot lu	*décomposition*	*Mot codé*	*indice*	*Mot ajouté au dictionnaire*
			0	""
"a"	"" + 'a'	(0,a)	1	"a"
"ab"	"a" + 'b'	(1,b)	2	"ab"
"r"	"" + 'r'	(0,r)	3	"r"
"abd"	"ab" + 'd'	(2,d)	4	"abd"
"b"	"" + 'b'	(0,b)	5	"b"
"abr"	"ab" + 'r'	(2,r)	6	"abr"
"abdg"	"abd" + 'g'	(4,g)	7	"abdg"

Pour décoder on effectue l'opération inverse.

1) Donner la déclaration du dictionnaire ?
2) Définir une fonction « *compresser* » permettant de transformer un texte selon l'algorithme ci-dessus.
3) Ecrire une fonction « *décompresser* » permettant de décoder un texte.
   Par exemple :
   (0,a) (1,b) (0,r) (2,d) (0,b) (2,r) (4,g) ... ➔ **aabrabdbabrabdg....**

Bonne Chance