IF3 Algorithmique et Programmation en langage C Durée : 2h

Examen de première session : Vendredi 1er juin 2007

Les documents, calculatrice, téléphone portable sont interdits
La qualité de l'écriture et de la présentation sera prise en compte dans la note finale.
Vous veillerez à respecter les notations et les règles d'écriture des algorithmes vues en cours et en TD. Soyez attentifs aux indices des tableaux et prenez garde à ne pas dépasser la fin des tableaux. Le barème est donné à titre indicatif.

Partie A : Questions de cours (3 pts)

- Qu'est-ce qu'une structure ? A quoi sert-elle ? Donnez un exemple simple de structure avec sa déclaration en algorithmique et en langage C.
- Qu'elle est l'utilité des fichiers en C ? Citez 3 opérations de base sur les fichiers.

Partie B : Algorithmique (8 pts)

Vous indiquerez clairement l'entête (ou la signature) de chaque algorithme que vous rédigerez (procédure ou fonction ainsi que les paramètres associés, leur type et leur catégorie).

1. (2 pts) Écrivez l'algorithme d'un sous-programme qui calcule et retourne un tableau contenant les 10 premiers termes de la suite U_n définie par :

$$U_0 = 1, U_{n+1} = \frac{U_n}{n+1}$$

2. (3 pts) On suppose avoir déclaré un tableau de 10 entiers T dont les valeurs ont été initialisées. Écrivez l'algorithme du sous-programme qui initialise un tableau d'entiers S, dont les éléments S[i] devront contenir la somme des éléments de T entre T[0] et T[i] (T[i] compris).

T :	2	7	0	3	1	6	4	5	9	8

S :	2	9	9	12	13	19	23	28	37	45

3. (3 pts) Un cercle est défini par les coordonnées de son centre (x,y) et son rayon R. Écrivez une structure de données permettant de stocker ces informations. En utilisant la structure décrite, écrivez l'algorithme d'une fonction qui testera si un point donné de coordonnées (abscisse, ordonnée) du plan est à l'intérieur, à l'extérieur ou sur le contour du cercle et qui renverra le code correspondant (-1, 1 ou 0) ainsi que la distance de ce point au centre du cercle.

T.S.V.P →

Partie C (Langage C) : séquences génétiques (9 pts)

Un *brin* d'ADN est formé de la répétition ordonnée des quatre bases appelées : *adénine*, *guanine*, *thymine* et *cytosine*. On écrit généralement une séquence d'ADN en juxtaposant les initiales des quatre bases :

G	Guanine	T	Thymine
A	Adénine	C	Cytosine

Du point de vue informatique, une séquence ADN peut être simplement représentée par une chaîne de caractères contenant les lettres A, G, T et C représentant les quatre bases de l'ADN. Exemple: char sequence_adn[MAX] = "ACGGTAGCTAGTTTCGACTGGAGGGGTA";

Dans cet exercice, vous allez vous entraîner à manipuler des séquences d'ADN à l'aide de chaînes de caractères.

1. (2 pts) Écrivez une fonction *bool adn_valide*(char sequence[MAX]) qui vérifie si la séquence passée en argument est bien une séquence ADN valide. La fonction devra retourner `true` si la séquence est composée uniquement de caractères A, G, T et C.

2. (2,5 pts) Écrivez, **sans utiliser la fonction `strcmp`**, une fonction qui compare deux séquences ADN et retourne `true` si les deux séquences sont similaires et `false` sinon. La fonction aura l'en-tête suivant: *bool compare(char seq1[MAX], char seq2[MAX])*. *Indice***:** si deux séquences n'ont pas la même longueur, il est clair qu'elles seront différentes. Il n'est dès lors pas utile de les comparer caractère par caractère.

Les bases de l'ARN sont les mêmes que pour l'ADN sauf la thymine (`T`) qui est remplacée par l'*uracile* (`U`).

3. (2 pts) Écrivez une fonction qui transcrit une séquence ADN en ARN, en remplaçant tous les `T` par des `U`. Utilisez l'en-tête de fonction suivante:
void adn_to_arn(char seq_adn[MAX], char seq_arn[MAX]).

On dit que l'adénine est complémentaire de la thymine et que la guanine est complémentaire de la cytosine.

4. (2,5 pts) Écrivez une fonction qui retourne le complémentaire d'une séquence ADN. Pour ce faire, il suffit de remplacer chaque base de la séquence par son complémentaire. Pour cette fonction, vous utiliserez le prototype suivant:
void complement(char seq_originale[MAX], char seq_complementaire[MAX])

Les mots du langage génétique, appelés *codons*, ont tous la même longueur. Ce sont des triplets pris parmi A, C, T(ou U) et G (dans n'importe quel ordre et sans doublons) qui servent à désigner des *acides aminés*.

5. (points bonus) Écrivez une fonction *recherche_codon* qui recherchera la présence d'un triplet bases (A, C, G, et T) dans une séquence ADN. La fonction retournera *true* si le codon est trouvé dans la séquence, *false* sinon. "codon" est forcément une chaîne de 3 caractères, contenant par exemple "ACT" ou "GAC" ou toute autre combinaison de 3 bases. Si le codon apparaît plusieurs fois dans la séquence d'ADN, vous devrez retourner le nombre d'occurrences du codon ainsi qu'un tableau contenant leurs positions.