REPUBLIQUE TUNISIENNE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Carthage
Institut National des Sciences Appliquées et de Technologie

Devoir Surveillé ☐	Examen ■	Session principale ■	
		Session de contrôle ☐	
Matière :	Algorithmique et structures de données II	**Semestre :** 2	
Enseignant(s) :	Majdi Jribi et Khaoula Bezzina	**Date:** 08 Juin 2021	
Filière(s) :	MPI	**Durée:** 1h30	
Barème :	11+9	**Documents :** autorisés	☐
Nombre de pages :	02	non autorisés	■

Toutes les réponses doivent être rédigées en langage C

Exercice 1

On dispose d'une série de mots de longueurs différentes. Pour cela on charge les mots dans un arbre binaire de recherche suivant leur longueur.
Un arbre binaire est représenté par la structure suivante :

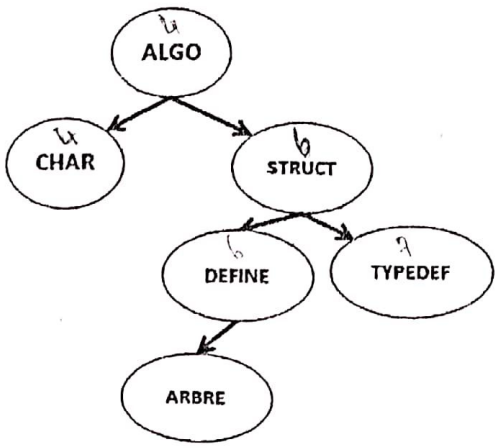
```
typedef struct arb {
char* mot;
structarb* fg;
structarb* fd;
}arb;
typedef arb* AB;
```

1. On suppose que les mots sont initialement dans un arbre binaire
   Ecrire la fonction **void min_max(AB racine, char* min, char* max)** qui cherche le mot le plus long et le mot le plus court dans l'arbre binaire.
2. Ecrire la fonction **void creerABR(AB abi, AB* abr)** qui permet de construire l'arbre binaire de recherche à partir de l'arbre binaire initial.

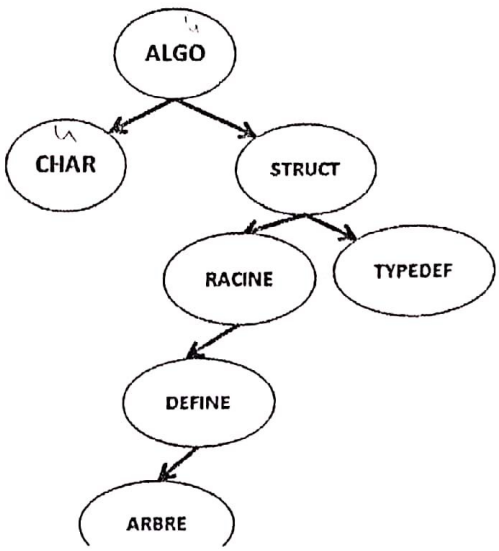
Pour étendre la série de mots, d'autres mots de longueurs déjà existantes dans l'arbre binaire de recherche doivent pouvoir y être insérés. Un mot de longueur existante doit être inséré comme fils gauche du premier nœud contenant un mot de même longueur.

Exemple : insertion du mot « RACINE » de longueur 6

Avant insertion



Après insertion



3. Ecrire la fonction **récursive** **void insere_mot(AB* racine, char* mot)** qui permet d'insérer un nœud dans l'arbre binaire de recherche avec la contrainte que si la longueur du mot existe, il sera placé comme déjà mentionné.

4. Ecrire la fonction **char* cherche_k_mot(AB racine, int longueur, Int k)** qui permet de retourner le **k**$^{\text{ième}}$ mot de longueur donnée de l'arbre binaire de recherche.

Exercice 2

On dispose d'un fichier« **logiciels.txt** » contenant la liste des logiciels utilisés par une entreprise. Chaque ligne de ce fichier contient les informations suivantes :

Code	: 5 Caractères numériques
Désignation	: 25 caractères
Langage de Programmation	: 20 caractères (représentant le nom du langage)
Taille en nombre d'octets	: 5 caractères numériques

1. Ecrire la fonction **void logicielsLangage(FILE* fp, char* langage)** qui permet d'afficher la liste des désignations de tous les logiciels d'un langage donné.

2. Ecrire la fonction **int updateLogiciel(FILE* fp, int code, char* nelleDesignation, char* taille)** qui permet de mettre à jour le fichier **« logiciels.txt »** en modifiant la designation et la taille d'un logiciel de code donné.
   La fonction devra retourner 1 si le logiciel a été modifié, 0 sinon.

3. Soit la structure de données suivante :

```
typedef struct langage {
char nom[22];
int totalOctets ;
} Langage ;
```

Le champ totalOctets représente la somme en octets de la taille des logiciels qui ont été crées par ce langage.

Ecrire la fonction **int statsLangage(Langage* t, FILE* fp)** qui permet de créer un tableau de Langage où chaque case du tableau contient un langage.

NB : Un langage est représenté par une seule case du tableau. Le nom d'un langage est son identifiant. La fonction devra retourner le nombre de langages représentés dans le fichier **« logiciels.txt »**.