EXAMEN

Semestre : 1 ☐ 2 ■

Session : Principale ■ Rattrapage ☐

Module : **Programmation Procédurale 2**
Enseignant(s) : **Equipe Prog. Proc. 2**
Classe(s) :**1A6,1A7,1A8,1A9,1A10**
Documents autorisés : OUI ■ NON ☐ Nombre de pages : 2
Calculatrice autorisée : OUI ☐ NON ■ Internet autorisée : OUI ☐ NON ■
Date : **17/05/2018** Heure **8h30** Durée :**90 min**

Une bibliothèque municipale souhaite informatiser son système de gestion d'emprunts d'ouvrages. L'ensemble des ouvrages de la bibliothèque seront sauvegardés dans un fichier binaire "biblio.bin" et l'ensemble des abonnées seront stockés dans un arbre binaire de recherche.

- Un ouvrage est caractérisé par : un code (entier) et un titre.
- Un abonné est caractérisé par : un numéro d'inscription, un nom et une liste simplement chainée des ouvrages empruntés.

1- Définir les types nécessaires**.**

Développer les fonctions qui permettent de :

2- Saisir un ouvrage :

Ouvrage Saisir_ouvrage() ;

3- Remplir le fichier binaire **nomfich** avec **n** ouvrages.

void Remplir_bibio(char nomfich[], int n);

4- Saisir un abonné. L'abonné n'a initialement aucun ouvrage.

void Saisir_Abonne(Abonne* x);

5- Insérer dans l'arbre binaire de recherche un abonné **x** selon son numéro d'inscription.

Abr Inserer_abonne(Abr A, Abonne x)

6- Chercher et retourner un ouvrage donné par son **code** à partir du fichier **nomfich**. Si l'ouvrage n'existe pas la fonction retourne un ouvrage avec un code = -1.

Ouvrage chercher_ouvrage(char nomfich[], int code);

7- Chercher et retourner l'adresse du nœud contenant l'abonné ayant le numéro **num.** Si l'abonné n'existe pas la fonction retourne NULL.

Nœud* chercher_Abonne(Abr A, int num);

8- Ajouter un ouvrage ***Ouv*** à la fin de la liste des ouvrages empruntés d'un abonné **x**.

Abonne Ajouter_ouvrage(Abonne x, Ouvrage Ouv) ;

9- Développer la fonction qui permet à un abonné donné par son numéro **num** d'emprunter un ouvrage donné par son code **code_ouv**. N.B : Utiliser les trois fonctions précédentes.

void Emprunter_ouvrage(Abr A, int code_ouv, int num) ;

10- Développer la fonction qui permet d'afficher les informations d'un abonné x

void Afficher_Abonne(Abonne x)

11- Développer la fonction récursive qui permet d'afficher les abonnés dans l'ordre décroissant de leur numéro d'inscription.

void Afficher_ABR (Abr Ar) ;

Grille de validation				
Algorithmique				
Déclaration des structures			**1.5 points**	
Saisir_ouvrage ()			**0.25 point**	
Remplir_biblio (char nomfich[],int n)			**2 points**	
saisir_Abonne (Abonne* x);			**0.5 point**	
Abr Insere_abonne(Abr A, Abonne x)			**1.5 point**	
Ouvrage chercher_ouvrage(char nomfich[], int code)			**1.5 point**	
Noeud* chercher_Abonne(Abr A, int num)			**1.5 point**	
Abonne Ajouter_ouvrage(Abonne x, Ouvrage Ouv)			**1.5 points**	
void Emprunter_ouvrage(Abr A, int code_ouv, int num,char nomfich[])			**2.25 point**	
void Afficher_Abonne(Abonne x)			**1 point**	
void Afficher_ABR (Abr A)			**1.5 point**	
Total algorithmique			**15 points**	
Exécution				
	Déclaration des variable/ Appel des fonctions / compilation		**Exécution (*)**	
menu		**1 point**		
Remplir_biblio		**0.5 point**		**0.5 point**
Saisir et ajouter abonné		**0. 5 point**		**0.5 point**
Emprunter un ouvrage		**0.5 point**		**0.5 point**
Afficher abonné		**0.25point**		**0.25 point**
Afficher abr		**0.25point**		**0.25 point**
Total Exécution				**5 points**
Sanctions				
La non utilisation de la programmation modulaire : **-1point**				
Modification des prototypes des fonctions : **-1point**				
L'utilisations des noms non significatifs pour les variables et structures : **-1point**				
Note totale /20				