


Institut Supérieur d'Informatique et de Mathématiques 	Année Universitaire : 2022-2023
	Examen de la session principale Matière : ASD 2 & Complexité Filières : L1 Info Enseignant : Sakka Rouis Taoufik

Exercice 1: (2+3+0.5 Points)

1. Sachant que les deux implémentations ci-dessous ont le même rôle, en utilisant le paradigme diviser pour résoudre, proposer une troisième implémentation pour cette fonction.

```

int Fonction1 ( int * t, int n) {
    if (n==1)
        return *t;
    else {
        if (Fonction1 (t, n-1) > *(t+n-1) )
            return Fonction1 (t, n-1);
        else
            return *(t+n-1);
    }
}

```

```

int Fonction2 ( int * t, int n) {
    int x;
    if (n==1)
        return *t ;
    else {
        x= Fonction2 (t, n-1);
        if (x > *(t+n-1) )
            return x ;
        else
            return *(t+n-1);
    }
}

```

2. Calculer la complexité de chacune de ces trois implémentations. Quelle est la fonction la plus performante (du point de vue complexité de calcul).

Exercice 2 : (4 Points)

Soit la déclaration suivant:

```

struct noeud {
    int info ;
    struct noeud * sag ;
    struct noeud * sad ;
};

```

On vous demande d'implémenter :

- Une fonction qui permet de calculer la somme des éléments d'un arbre d'entier.
- Une fonction qui permet de chercher la hauteur d'un arbre d'entier.

Exercice 3 : (10.5 Points)

Afin d'améliorer le temps de recherche d'un élément dans un ensemble d'éléments, on souhaite matérialiser la structure de données "TabListe" définie comme suite :

```
struct cellule {  
    int cle;  
    struct cellule *suivant;  
};  
  
struct liste {  
    struct cellule *premier;  
    struct cellule *dernier;  
};  
const unsigned N = 10;  
struct liste TabListe [N];
```

Pour cela, on vous demande d'implémenter les fonctions C suivantes :

- **void CreerTabListe ()** : permettant d'initialiser la structure de données TabListe avec des listes vides. (1 p)
- **unsigned TabListeVide ()** : permettant de vérifier si la structure de données TabListe est vide ou non. (1 p)
- **void AjouterElement (int x,)** : permettant d'ajouter un élément dans la structure de données TabListe. L'indice de la liste à laquelle cet élément sera ajouté est déterminé par l'application d'une opération particulière sur l'entier x. Il est important de noter que l'opération particulière à utiliser pour calculer cet indice n'est pas encore spécifiée. (3 p)
- **struct cellule* Rechercher (int x,)** : qui retourne l'adresse de la première cellule contenant la valeur x (NULL si x n'est pas présent dans la structure de données). (2 p)
- **void SuppElement (int x,)** : qui supprime la première cellule contenant l'entier x dans la structure de données TabListe. Vous pouvez utiliser la fonction précédente "Rechercher" ainsi que les fonctions de suppression vues en cours. (2 p)
- **struct liste* Transvider ()** : qui permet de copier les éléments de la structure de données TabListe dans une liste simplement chaînée. Pour ce faire, vous devez parcourir la structure de données TabListe (liste par liste) et copier les éléments de chacune d'entre elles dans la nouvelle liste. (1.5 p)

NB. Vous pouvez appeler les fonctions vues dans le chapitre Liste.