Travail à rendre

(à rendre avant le 28/11/2025 à 00h00)

EXERCICE 1 :

L'objectif de ce problème est d'écrire un programme qui permet de faire la somme de deux grands nombres entiers.

Pour réaliser cet objectif on doit lire une chaîne de caractères représentant un grand nombre entier et le charger par paquets de 4 chiffres dans une pile.

L'élément de la pile contiendra un champ ***Paquet*** de type entier qui représente les 4 chiffres du paquet et un pointeur ***Precedent***.

```
typedef struct ElementPile {
        int Paquet;
        struct ElementPile *Precedent;
} Element;
```

Chaque nombre est donc représenté par une pile dynamique, et pour avoir le contrôle de la pile, il est préférable de sauvegarder les éléments suivants :

- Le pointeur ***Sommet*** contiendra l'adresse du premier élément qui se trouve au sommet de la pile.
- La variable entière ***Npaquets*** contiendra le nombre d'éléments (nombre de paquets).

Pour simplifier l'écriture du programme et pour ne pas avoir plusieurs variables globales, une autre structure de donnée doit être ajoutée (obligatoire) pour représenter les nombres à additionner (**Nombre1** et **Nombre2**) et le nombre somme (**Nombre3**) :

```
typedef struct {
        Element *Sommet;
        int Npaquets;
} GrandNombre;
```

Les variables globales du programme peuvent être :

```
GrandNombre Nombre1, Nombre2, Somme;
char ch1[30];  // ch1 chaîne qui représente le premier grand nombre
char ch2[30]; // ch2 chaîne qui représente le deuxième grand nombre
```

1) Écrire les fonctions ***initialiser***, ***pileVide***, ***pilePleine***, ***empiler*** et ***depiler*** associées à une pile.
   N.B : *paramétrer ces fonctions pour qu'elles soient appelées par plusieurs piles.*
2) Écrire la fonction ***remplir(char *ch, GrandNombre *N)*** qui permet de lire et charger une chaîne de caractères représentant le grand nombre par paquets de 4 chiffres dans la pile d'entiers.
3) Écrire la fonction ***somme(GrandNombre *N1, GrandNombre *N2, GrandNombre *S)*** qui permet d'additionner deux grands nombres chargés chacun dans une pile.
4) Écrire la fonction ***afficher(GrandNombre *N)*** qui permet d'afficher correctement un grand nombre entier chargé dans une pile d'entiers.
5) Écrire la fonction ***main()*** dont on trouve la lecture des deux grands nombres (***ch1*** et ***ch2***) et l'affichage du résultat de leur somme.

EXERCICE 2 :

On considère la structure de donnée suivante qui permet de représenter un nœud d'un arbre binaire de recherche (ABR) d'entiers :

```
typedef struct element{
    int donnee ;
    struct element *sag;
    struct element *sad;
}Noeud;
```

1) Donner les fonctions nécessaires pour manipuler un arbre binaire de recherche (ABR) :
   a) Initialisation de l'arbre.
   b) Préparation d'un nœud.
   c) Ajout d'un nœud.
2) Implémenter des fonctions assurant les trois types de parcours en profondeur (Préfixe, Infixe et Postfixe).
3) Ajouter une fonction ***void remplirTab(Noeud *ar, int *T, int *N)*** qui permet de remplir les données d'un arbre dont on fournit l'adresse de la racine ***ar***, dans un tableau ***T*** de taille ***N***.
4) Ajouter une fonction ***void Recherche(Nœud *ar, int *min, int *max)*** permettant la recherche du minimum et du maximum de l'arbre binaire de recherche dont la racine est pointée par ***ar***.
5) Rédiger la fonction ***main()*** pour tester les fonctions des questions précédentes.