Exercice 1 :

Écrire une fonction en **C** avec l'entête suivante :

```
#define k 5
int tous_egaux(unsigned int *t);
```

- La fonction prend en paramètre un tableau `t` contenant `k` nombres positifs.
- Elle retourne `1` si **tous les éléments du tableau sont égaux**, et `0` sinon.

Tester votre fonction depuis le main:

- Avec un tableau statique.
- Avec un tableau alloué dynamiquement.

Exercice 2:

La fonction "free" permet de libérer un espace mémoire (que nous supposons, dans cet exercice, composé des entiers) pointé par p et qui a été préalablement alloué de manière dynamique. Cependant, la fonction free ne met pas la valeur du pointeur p à NULL.

- Ecrire une fonction, appelée liberer, qui permet de libérer un espace mémoire pointé par p et qui réinitialise la variable p à NULL. La fonction ne retourne aucune valeur.
- Tester votre fonction depuis la fonction mail.

Indications : Le travail demandé consiste à compléter le programme suivant :

```
#include <stdio.h>
#include <stdlib.h>
#define Taille 2000

void liberer (...){
    ......
}

int main (void)
{
    void libérer (....);
    int *p=(int *) malloc (Taille*sizeof (int));
    liberer (....);
    printf ("La nouvelle valeur du pointeur est : %p \n", p);
    return 0;
}
```

Exercice 3:

On a demandé à un étudiant d'écrire un programme C qui

- affecte les valeurs 10 et 20 à deux variables,
- puis affiche le contenu de ces deux variables.

L'étudiant, voulant à tout prix n'utiliser que les pointeurs, a écrit le programme suivant :

```
#include <stdio.h>
int main (void)
{
unsigned int *a, *b;
*a=10;
*b=20;
printf ("Après affectation : A=%u, B=%u \n", *a,*b);
return(0);
}
```

Sans surprise son programme ne fonctionne pas!

- Expliquer pourquoi son programme ne fonctionne pas.
- Corriger son programme en utilisant que des variables de type pointeurs.

Exercice 4:

- Déclarer une constante k (par exemple k est égale à 5) grâce à la directive #define.
- Écrire une fonction, appelée fusion, qui prend en paramètres deux tableaux (de même taille k) triés (de manière croissante) et fusionne ces deux tableaux dans un troisième tableau passé aussi en paramètre. Le troisième tableau résultat (de taille 2xk) doit-être également trié.
- Tester votre fonction depuis le programme principal main.

Exercice 5 :

1. Ecrire une fonction C qui a l'entête suivante :

   ```
   unsigned char contient_deux_opposes (int tab [], int n)
   ```

   où :

   tab : est un tableau d'entiers relatifs différents de 0 (qui peuvent être soit positifs soit négatifs), et

   n : est la taille du tableau.

   La fonction *contient_ deux_ opposes* retourne la valeur 1 s'il existe deux nombres opposés dans le tableau; c'est-à-dire si le tableau contient deux entiers différents $T[i]$ et $T[j]$ (avec $0 \leq i < n$ et $0 \leq j < n$) tel que :

   $$T[i] + T[j] = 0.$$

   La fonction retourne 0 dans le cas contraire.

2. Tester votre fonction *contient_ deux_ opposes* depuis le main.
3. Supposons maintenant que le tableau **Tab**, passé en paramètre de la fonction, est trié par ordre croissant. Proposer une version efficace de votre fonction *contient_ deux_ opposes* (en $\mathcal{O}(n)$).

Exercice 6 :

Ecrire une fonction C qui a l'entête suivante :

```
#define k 5
int tous_differents (unsigned int *t)
```

Cette fonction prend en paramètre un tableau de k nombres postifs et retourne 1 si toutes les nombres dans t sont différents. La fonction retourne 0 dans le cas contraire.

Exercice 7 :

- Ecrire une fonction **récursive** qui :
  - prend en paramètre un tableau d'entiers (tous positifs sauf le dernier élément qui est égal -1) et
  - retourne la somme des entiers positifs qui le composent.
- Réécrivez la fonction précédente sans utiliser les symboles [] utilisés dans les tableaux.
- Tester votre fonction depuis le main, en utilisant dans un premier temps un tableau, puis une allocation dynamique.

Exercice 8.

- Que fait le programme suivant :

```
#include <stdio.h>
/*
Que fait ce programme ?
 */
int main() {
    char a='a';
    char *p=&a;
```

```
    char *adresse=(char *) 0x0;
    while (adresse != p)  adresse++;
    printf (".... est %p \n", adresse);
    return 0;
}
```

- Compiler et exécuter le programme ci-dessus.
- A votre avis, quel est l'objectif de cet exercice ?