Exercice 1 :

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 2 :

- 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 3:

- Écrire une fonction en C qui prend en paramètres deux entiers positifs `a` et `b`, et retourne `1` si `a` est stricteemnt plus grand que `b`, `-1` si `a` est strictement plus petit que `b`, et `0` s'ils sont égaux.

  Exemples :
  - Si `a = 16` et `b = 15`, la fonction retourne `1`.
  - Si `a = 20` et `b = 30`, la fonction retourne `-1`.
  - Si `a = 18` et `b = 18`, la fonction retourne `0`.
- Reprendre l'exercice précédent, mais cette fois-ci, aucun opérateur arithmétique (`+`, `-`, etc.) ni de comparaison (`==`, `>`, `<`, `>=`, `<=`, `!=`) ne doit être utilisé.

Exercice 4:

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 5:

- 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 6 :

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 7

Ecrire une fonction récursive qui vérifie si un tableau est trié de manière croissante. Cette fonction a l'entête suivante :

```
int sorting_checking_recursive(int tab[], int n)
```

Exercice 8:

Écrire une fonction en langage C pour déterminer si le problème du sac à dos de Dora, tel que vu en cours, possède une solution. Si une solution existe, la fonction devra l'afficher et retourner 1 ; dans le cas contraire, elle retournera 0.

Les données du problème sont les suivantes :

- Un sac à dos d'une capacité de t grammes.
- Un tableau d'entiers *tab* (où chaque entier représente le poids, en grammes, de pierres précieuses), terminé par -1 (indiquant qu'il ne s'agit pas d'une pierre précieuse).