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.

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:

- Déclarer une constante k (typiquement k est égale à 3) grâce à la directive #define.
- Écrire une fonction, appelée Tictactao_gagnant, qui vérifie si un tableau à deux dimensions correspond à une configuration de tic tac tao gagnante.
- Tester votre fonction depuis le programme principal main.

Pour rappel, une configuration de tic tac tao gagnante est vraie lorsqu'il existe une ligne (resp. une colonne, une diagonale) qui contient k croix ou k cercles.

Exercice 5:

1. Dans une fête foraine, un exposant propose le jeu suivant et dont la partie coûte deux euros. Dans le stand de l'exposant, on trouve :
   - Un boulier qui contient toutes les boules possibles à deux chiffres.
   - Un tableau de gains :

–	–	–	–	–	–	–	–	–	9 1€
10 100€	11 100€	12 100€	13 100€	14 100€	15 100€	16 100€	17 100€	18 1€	19 100€
20 100€	21 100€	22 100€	23 100€	24 100€	25 100€	26 100€	27 1€	28 100€	29 100€
30 100€	31 100€	32 100€	33 100€	34 100€	35 100€	36 1€	37 100€	38 100€	39 100€
40 100€	41 100€	42 100€	43 100€	44 100€	45 1€	46 100€	47 100€	48 100€	49 100€
50 100€	51 100€	52 100€	53 100€	54 1€	55 100€	56 100€	57 100€	58 100€	59 100€
60 100€	61 100€	62 100€	63 1€	64 100€	65 100€	66 100€	67 100€	68 100€	69 100€
70 100€	71 100€	72 1€	73 100€	74 100€	75 100€	76 100€	77 100€	78 100€	79 100€
80 100€	81 1€	82 100€	83 100€	84 100€	85 100€	86 100€	87 100€	88 100€	89 100€
90 1€	91 100€	92 100€	93 100€	94 100€	95 100€	96 100€	97 100€	98 100€	99 1€

- Pour connaître son gain, on tire au hasard un nombre dans le boulier. Puis on enlève (soustrait) à ce nombre les chiffres qui le compose. Le gain associé au nombre obtenu est donné par le tableau des gains. Par exemple, si le boulier donne 42. On enlève à ce nombre les chiffres 4 et 2. On obtient 36. Le gain est de 1 euro.
- Un enfant a essayé plusieurs fois ce jeu sans jamais gagner. Il se demande s'il y a une stratégie gagnante à ce jeu. Il sollicite votre aide. Pouvez-vous écrire une fonction C pour répondre à sa question ?
- NB. Le tableau a une propriété très simple et facile à découvrir.

Exercice 6.

- 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 ?

Exercice 7:

Un étudiant a écrit le programme suivant :

```
#include <stdio.h>
void echange_adr (unsigned int *a, unsigned int *b){
  unsigned int *c;
  *c=*a;
  *a=*b;
  *b=*c;
}

int main (void){
  void echange_adr (unsigned int *a, unsigned int *b);
  unsigned int x, y;
  unsigned int *p=&x, *p1=&y;
  x=5;
  y=10;
  echange_adr(p, p1);
  printf ("Grâce aux pointeurs, après l'appel à la
```

```
        fonction echange_adr x=%u et y=%u.\n", x,y);
  return(0);
}
```

Il pense que son programme n'est pas juste. Pouvez-vous l'aider?

Exercice 8:

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 9:

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;
}
```