❑ Exercice 1 : *Déclarations*

1. Ecrire les instructions permettant de :
   a) Déclarer une variable `n` de type entier.
   b) Déclarer une variable `x` de type flottant initialisée à 1.
   c) Déclarer une variable `test` de type booléen, quelle librairie est nécessaire ?
   d) Déclarer un tableau de 5 entiers initialisés aux valeurs $\{1, 4, 9, 16, 25\}$
   e) Déclarer une chaine de caractères initialisée à « *Trop bien le langage C* » ;
   f) Déclarer une variable de type `char` contenant le caractère \$.
2. Ecrire les signatures des fonctions suivantes :
   a) `divisible_par` qui prend en argument deux entiers n et p et renvoie un booléen.
   b) `somme` qui prend en argument un tableau de flottant et un entier et renvoie un flottant.
   c) `carre` qui prend en argument un entier renvoie un entier.
   d) `affiche` qui prend en argument un booléen et ne renvoie rien.

❑ Exercice 2 : *boucles*

Indiquer si les boucles `for` suivantes sont correctes ou non et si elles le sont, donner les valeurs prises par la variable d'itération.

1. `for (int m = 1; m < 10; m=m+1)`
2. `for (int i = 12; i < 10; i++)`
3. `for (int k = 0; k < 10; i--)`
4. `for (j = 0; j < 10; j++)`
5. `for (int i = 42; i > 21; i--)`

❑ Exercice 3 : *Comportements indéfinis*

1. Rappeler ce qu'est un *comportement indéfini* en C.
2. Donner au moins un exemple de programme ayant un comportement indéfini.
3. On considère la fonction suivante en C :

```c
int signe(int n)
{
    if (n>0)
    {
        return 1;
    }
    if (n<0)
    {
        return -1;
    }
}
```

et on précise que le standard du langage C indique que :

> « *If control reaches the closing curly brace (}) of a non-void function without evaluating a return statement, using the return value of the function call is undefined behavior.* »

En déduire un cas où l'utilisation de cette fonction produit un comportement indéfini et écrire une fonction `main` appelant la fonction `signe` et produisant un comportement indéfini.
Remarque : lors de la compilation, on obtient l'avertissement suivant : « *control reaches end of non-void function* »

❑ Exercice 4 : *Opérations sur les types de bases*

1. On considère le programme suivant :

```c
#include <stdio.h>

int main()
{
    float r = 5 % 3;
    printf("Résultat = %f\n", r);
}
```

a) Quel est le résultat produit ? Expliquer.
b) Que se passe-t-il si on change la ligne 5 en `float r = 5.0 % 3.0;`

2. Ecrire une expression booléenne valant vraie si `a` est égale à `b+c` ou si `c` est non nul et que `a` est divisible par `c`. L'évaluation de cette expression génère-t-elle une erreur si `c` est nul ?
3. On suppose déjà déclarée la variable `char c = 'A';`. Quel est le résultat de l'instruction suivante : `printf("%c \n",c+1);`

❑ Exercice 5 : *Portée*

On considère le programme C suivant :

```c
#include <stdio.h>

const float pi = 3.1415;
int k = 1;

int main() {
  float s = 0;
  int k = 1;
  while (pi * pi / 6 - s > 0.25) {
    float v;
    v = 1.0 / (k * k);
    s += v;
    k = k + 1;
  }
  return 0;
  }

```

1. Pour chacune des variables du programme, indiquer si elle est globale ou locale et donner sa portée.
2. Déterminer la valeur de chacune des variables existantes juste avant l'instruction `return` de la ligne 15.
   ⛑ On peut utiliser une calculatrice !

❑ Exercice 6 : *Conversion*

Déterminer le type et la valeur des expressions suivantes. Indiquer lorsqu'une conversion implicite ou explicite a eu lieu.

1. `!(5<7)`
2. `3 + 0.14`
3. `(int)7.5 + (int)12.3`
4. `7.0 /2`
5. `(true || false) && (false || true)`
6. `(int) 19.6 % 4`

❑ Exercice 7 : *Analyser un programme*

On considère le programme suivant :

```c
#include <stdio.h>

void echange(int a, int b)
{
    int temp = a;
    a = b;
    b = temp;
}

int main()
{
    int a = 12;
    int b = 50;
    echange(a, b);
    printf("a = %d\n", a);
    printf("b = %d\n", b);
    return 0;
}
```

1. Quel sera le résultat de l'exécution de ce programme ? Pourquoi ?
2. Quel sera l'affichage produit si on déplace l'affichage des variables `a` et `b` dans la fonction `echange` ? Pourquoi ?

❑ **Exercice 8** : *Programmes à commenter*

Commenter les programmes suivants en précisant les résultats obtenus (si le programme compile) ou les erreurs ou avertissements éventuels et les comportements indéfinis.

- Programme A :

```c
#include <stdio.h>

int main()
{

    int tab[5] = {42};
    for (int i = 0; i < 5; i++)
    {
        printf("%d \n", tab[i]);
    }
}
```

- Programme B :

```c
#include <stdio.h>
int main()
{
  int tab[5] = {42};
  int i = 0;
  while (i < 6)
  {
    printf("%d \n", tab[i]);
    i = i + 1;
  }
}
```

- Programme C :

```c
#include <stdio.h>

int main()
{
    float s = 0;
    for (int i = 1; i < 10000; i++)
    {
        s = s + 1 / i;
    }
    printf("somme =%f\n", s);
}
```

- Programme D :

```c
#include <stdio.h>

int fonction(int a)
{
    a = a + 1;
    printf("Valeur de a = %d\n", a);
}

int main()
{
    int c = 2;
    int d = fonction(c);
    printf("valeur de c = %d \n", c);
    printf("valeur de d = %d \n", d);
}
```

❑ **Exercice 9** : *A propos de* ++

1. Rappeler le type auquel il s'applique et le rôle de l'opérateur ++
2. On suppose déjà déclarée une variable `int a = 42;`.
   Que vaut b si on a écrit `int b = a++ + a;` ?
   Détailler votre raisonnement.