1 Échauffement sur les pointeurs

Question 1 : Déclarez une variable entière a et affichez sa valeur sans l'avoir initialisée. Créez ensuite un pointeur p qui contient l'adresse de a et modifiez la valeur de a à partir de p. Vous afficherez alors la valeur de a, pour vérifier que celle-ci a bien été modifiée.

Question 2 : Reprenez la question précédente et affichez aussi la valeur de p.

Question 3 : Recopiez et exécutez le programme de la Figure 1.

```c
#include<stdio.h>
#include<stdlib.h>

void plus_un(int a){
    a++;
}
int main(int argc, char *argv[]){
    int a;
    a=0;
    printf("a=%d\n",a);
    plus_un(a);
    printf("a=%d\n",a);
}
```

FIGURE 1 –

Vous remarquerez que la valeur de a ne change pas, alors que la fonction "plus un" (+1) a été appelée, pourquoi ? Corrigez cela en ne faisant que 3 petites modifications du code. Votre programme affichera alors

```
a=0
a=1
```

2 Tableaux dynamiques à deux dimensions

Question 1 : Écrivez un programme dont le `main` contiendra la déclaration d'un tableau à 2 dimensions dont vous initialiserez toutes les cases à 0.

Les nombres de lignes et de colonnes seront pris comme arguments dans la ligne de commande, par exemple

```
$ ./mon_programme 10 5
```

correspond à la création d'un tableau de 10 lignes et 5 colonnes.

Question 2 : Codez une fonction pour afficher le tableau et une pour le remplir avec des valeurs aléatoires. Vous pourrez alors vérifier, dans l'esprit de l'exercice précédent, que le tableau avant et après modification est différent.

Astuce : voici comment obtenir des valeurs aléatoires en C :
- importez la bibliothèque `time.h`
- initialisez la graine avec la commande `srand(time(NULL))`
- maintenant, chaque appel à `rand()` vous donnera un entier *pseudo*-aléatoire [1].

1. Attention, ces entiers sont en général très grands, utilisez `rand()%10` si vous voulez juste des nombres entre 0 et 9. En toute rigueur, ces nombres ne sont pas distribués uniformément entre 0 et 9, mais ce problème dépasse largement le niveau de ce cours, donc ne vous en souciez pas.

3 Attention à la fuite...

Joint à ce TP vous trouverez un programme `birthday.c` qui une fois compilé s'exécute comme ceci :

```
$ ./birthday 10 1000
```

Le programme vous donnera alors la fréquence avec laquelle une classe de 10 élèves admet au moins deux élèves avec exactement la même date d'anniversaire ; dans ce contexte, on dit qu'on a trouvé une *collision*.

Le deuxième paramètre (ici 1000) donne le nombre de fois que l'on souhaite réaliser l'expérience ; donc plus ce nombre est grand, plus la fréquence donnée sera proche de la probabilité théorique.

Question 1 : Analysez et comprenez le code de ce programme. Je vous rassure il est très simple, il vous suffit de vous rappeler qu'une probabilité est obtenue en divisant le nombre total de succès (ici le nombre de collisions) sur le nombre total d'expériences.

Question 2 : Exécutez le code avec pour paramètres $(50, 5\ 000\ 000)$ puis, si votre ordinateur est toujours vivant, avec $(50, 10\ 000\ 000)$. Regardez ce qui se passe pendant l'exécution du programme avec la commande Unix `top` que vous exécuterez dans un autre terminal.

Question 3 : Corrigez le problème identifié à la question précédente. Si vous l'aviez déja identifié à la question 1, bravo !