Laboratoire 13

- **Buts**
  - Exercer les structures de contrôle et initialisation à la simulation numérique.
- **Travail à réaliser**
  - Exercice 3.32 : Estimation de π. La bibliothèque `<random>` met à disposition divers outils pour la génération de nombres aléatoires. Le programme suivant donne des exemples simples de leur utilisation.

```
#include<iostream>
#include<random>
#include <functional>
int main()
{
  auto gen_double01 = std::bind(std::uniform_real_distribution<double>(0,1), std::mt19937(1234)); // germe: 1234
  auto gen_int1_3 = std::bind(std::uniform_int_distribution<int>(1,3), std::mt19937(987));  // un autre germe
  std::cout << gen_double01() << " " << gen_int1_3() << std::endl;
}
```

  - Exercice 3.33 : Le problème des 3 portes
- **Délai**
  - Fin de la séance

Laboratoire 14

- **Buts**
  - Initialisation aux problèmes numériques et à l'algorithmique
- **Travail à réaliser**
  - Exercice 3.28 : Calcul de la série harmonique $\sum_{i=1}^{n} 1/i$. Faire tout d'abord le calcul avec des `float`, soit en itérant de 1 à $n = 10^7$, soit en itérant dans l'autre sens. Recommencer avec le type `double`.
  - Exercice 3.29 : Calcul du plus petit commun multiple.
- **Délai**
  - Fin de la séance

Laboratoire 15

- **Buts**
  - Implantation de fonctions simples
- **Travail à réaliser**
  - Implanter une fonction qui retourne le plus grand diviseur commun entre deux nombres entiers passés en paramètre. Utiliser l'algorithme d'Euclide `https://fr.wikipedia.org/wiki/Algorithme_d%27Euclide` pour réaliser cette implantation.
  - Implanter la fonction d'exponentiation modulaire $b^e \mod m$, où b, e et m sont des entiers positifs. Pour implanter cette fonction efficacement, on peut remarquer que si e est pair, sa valeur vaut $((b^2) \mod m)^{(e/2)}$, ce qui permet de diviser par 2 le nombre de multiplications. Si b est impair, sa valeur vaut : $b \cdot b^{e-1} \mod m$. On en dérive l'algorithme efficace donné ci-dessous, à implanter sous la forme d'une fonction.
- **Délai**
  - Fin de la séance

Exponentiation modulaire

Input: $b, e, m \in \mathbb{N}$
Result: $r = b^e \mod m$

$r \leftarrow 1$
while $e > 0$ **do**
  if $e \mod 2 = 0$ **then**
    $b \leftarrow b^2 \mod m; e \leftarrow e/2$
  else
    $r \leftarrow r \cdot b \mod m; e \leftarrow e - 1$
end

Laboratoire 16

- **Buts**
  - Implanter la méthode de cryptographie à clé publique de Clifford Christopher Cocks

Processus d'échange de message secret avec un canal non sécurisé

Si Bob veut transmettre un message secret à Alice, il demande à cette dernière une clé publique. Pour cela, Alice génère deux nombres premiers p et q ainsi qu'un nombre premier $e < (p-1)\cdot(q-1)$. Elle transmet à Bob la valeur de e et celle de $n = p \cdot q$ (qui forment la clé publique, sans révéler p et q qui doivent rester secrets). Elle lui demande de chiffrer son message $m < n$ en calculant $c = m^e \mod n$ et de lui transmettre la valeur de c. Pendant ce temps, Alice calcule $d = e^{(p-1)\cdot(q-1)-1} \mod ((p-1)\cdot(q-1))$ qui est l'inverse de e modulo n. En calculant $m = c^d \mod n$ elle peut donc retrouver le message secret de Bob.

- **Travail à réaliser (individuellement)**
  - Implanter le processus de cryptographie à clé publique.
  - Le programme devra demander à l'utilisateur, avec vérification, trois nombres premiers p, q et e $(< 2^{31})$
  - Il devra ensuite calculer d et vérifier que $(m^e \mod n)^d \mod n \equiv m \quad \forall m < n$, autrement dit, que l'on arrive bien à retrouver univoquement le message d'origine à partir du message crypté.
- **Délai**
  - Fin de la séance suivante. Le laboratoire sera noté.