Exercice 1. Écrire un programme qui lit en entrée deux entiers a et b, et qui affiche les entiers de a à b inclus dans l'ordre croissant si $a \leq b$, et de b à a inclus sinon.

Exercice 2. Reprendre l'exercice précédent, sans utiliser de boucles.

Exercice 3. Écrire une fonction prenant en entrée deux bornes $a \leq b$ et tirant un entier aléatoire uniforme dans l'intervalle $[\![a, b]\!] = \{a, a+1, \ldots, b\}$.

Exercice 4. Écrire un programme lisant en entrée un entier n, puis n entiers quelconques $x_0, \ldots x_{n-1}$ et affichant leur somme.

Exercice 5. Écrire une fonction déterminant si un nombre est premier. En l'utilisant, écrire un programme qui prend en entrée un entier n et affiche les n premiers nombres premiers.

Exercice 6. En s'inspirant de l'exponentiation rapide, écrire une fonction calculant le produit de deux nombres sans jamais utiliser l'opérateur de multiplication, **sauf** pour faire des multiplications par 2.

Exercice 7. Un nombre premier tronquable est un nombre premier qui reste premier si l'on enlève un nombre arbitraire de chiffres à sa droite. Par exemple, 7393 est un nombre premier tronquable car $7393, 739, 73$ et 7 sont tous premiers. A l'aide d'un programme, déterminer le plus grand nombre premier tronquable inférieur à 10^9.

Exercice 8. La conjecture de Goldbach affirme que tout entier pair supérieur à 3 peut s'écrire comme la somme de deux nombres premiers. Vérifier que cette conjecture tient pour tous les entiers inférieurs à 10^6.

Exercice 9. Écrire une fonction calculant et affichant la décomposition en facteurs premiers d'un entier n, puis écrire un programme permettant de l'utiliser via le terminal.

Exercice 10. Corriger le code suivant :

```c
int main(){
  // Lire N dans le terminal
  int N;
  scanf("%d\n", N);
  // générer et afficher un nombre aléatoire entre 1 et N inclus
  int x = rand()%N;
  printf("%d\n", x);
}
```

Exercice 11. La méthode de Monte-Carlo d'approximation de π consiste à tirer des points aléatoires dans le carré unité $[0, 1] \times [0, 1]$, et à compter la proportion p de points qui sont dans le quart de cercle de centre 0 et de rayon 1. L'aire du carré étant 1, et l'aire du quart de cercle étant $\frac{\pi}{4}$, on s'attend à avoir $4p \approx \pi$. Implémenter cette méthode en C, et approximer π en tirant 10^8 points. L'approximation est-elle précise ?

Exercice 12. On suppose disposer d'une fonction `float f(int n)`, dont le temps d'exécution est très long. Expliquer pourquoi le programme suivant, qui calcule la valeur maximale atteinte par f entre 1 et 1000, n'est pas efficace, et le corriger :

```
int main(){
  float m = f(1); // maximum courant
  for (int i = 2; i <= 1000; i++){
    if (f(i) > m){
      m = f(i);
    }
  }
  printf("%f\n", m);
}
```

Exercice 13. Déterminer à la main ce qu'affiche le programme suivant lorsqu'on le compile et qu'on l'exécute.

```
#include <stdio.h>

int main(){
  int a = 0;
  int b = 1;
  int s = 0;
  while (a < 200){
    s += b;
    a += 1;
    b += 2;
  }
  printf("%d\n", s);
}
```

Exercice 14. Écrire une fonction qui affiche l'heure actuelle **aussi précisément que possible**. L'utiliser pour écrire un programme horloge qui affiche l'heure chaque seconde.

Exercice 15. Écrire un programme qui génère en boucle des problèmes de calcul mental aléatoires et demande à l'utilisateur de les résoudre. On pourra générer des additions, des multiplications, mais aussi des questions plus exotiques : des calculs de fractions, de racines carrées à 3 décimales près, des résolutions d'équations, etc...

Exercice 16. (*Difficile*) On plie un ruban de papier en deux, k fois d'affilées, en ramenant systématiquement le bord droit sur le bord gauche, puis on le déplie. On voit apparaître des pliures orientées vers le haut (H), et d'autres vers le bas (B). Par exemple, pour $k = 1$, une seule pliure H apparaît, et pour $k = 2$, trois pliures apparaissent : HHB (voir schéma).

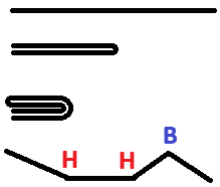

FIGURE 1 – Pliage et dépliage pour $k = 2$

Écrire un programme prenant en entrée k et affichant le sens des pliures de gauche à droite. On pourra chercher une structure récursive au problème, et commencer par déterminer le **nombre** de pliures qui apparaissent.