À plusieurs points dans le sujet il est nécessaire de savoir générer un entier tiré au hasard uniformément, on peut obtenir un tel nombre aléatoire r compris entre 0 (inclus) et n (exclus) avec la fonction `rand` :

```
r = rand() % n;
```

Cette fonction est inclue dans la bibliothèque standard : vous devrez donc ajouter `#include <stdlib.h>` au début de votre programme.

Exercice 1 : Dessins

Q. 1 Écrire un programme qui affiche un rectangle de taille 6×5 composé uniquement d'astérisques (`*`). Le résultat attendu est :

```
*****
*****
*****
*****
*****
*****
```

Q. 2 Écrire un programme qui affiche un rectangle de taille 6×5 composé uniquement d'astérisques (`*`) et de plus (`+`) formant des diagonales qui s'alternent. Le résultat attendu est :

```
*+*+*
+*+*+
*+*+*
+*+*+
*+*+*
+*+*+
```

Exercice 2 : Quelques suites numériques

1. Nombres aléatoires premiers entre eux

Dans cet exercice, on s'intéresse à estimer numériquement la probabilité que deux entiers i et j de l'intervalle $[\![1, n]\!]$ soient premiers entre eux.

Q. 1 Définir une fonction `int premiers_entre_eux(int a, int b);` retournant vrai (1) si et seulement `a` et `b` sont premiers entre eux.

Q. 2 Définir une fonction `float proba_premiers_entre_eux(int n);` donnant la probabilité que deux entiers i et j de l'intervalle $[\![1, n]\!]$ soient premiers entre eux.

Q. 3 Expérimenter avec la valeur asymptotique de la suite $\sqrt{\frac{6}{\texttt{proba\_premiers\_entre\_eux(n)}}}$

Q. 4 Lorsque `n` devient trop grand il n'est plus possible de tester toutes les paires de l'intervalle $[\![1, \texttt{n}]\!]^2$. Définir une fonction `float proba_approx_premiers_entre_eux(int n, int m);` choisissant `m` paires d'entiers au hasard dans $[\![1, n]\!]$ et calculant l'approximation observée sur cet échantillon de la probabilité que deux entiers soient premiers entre eux.

2. Sommes dites “de Riemann”

Q. 5 Définir une fonction `float mystere(int n);` calculant la somme suivante :

$$S_n = 4\sum_{k=1}^{n} \frac{n}{k^2+n^2}$$

Expérimenter avec la valeur asymptotique (quand n est grand) de la suite.

Q. 6 Définir une fonction `float mystere_stable(float eps);` calculant la somme S_N pour le premier indice N tel que $|S_N - S_{N-1}| < \texttt{eps}$.

3. Itérations vers un point fixe

On s’intéresse dans cette partie aux suites définies de la manière suivante :

$$\begin{cases} u_0 & = \texttt{c} \\ u_{n+1} & = f(u_n) \end{cases}$$

Q. 7 Définir une fonction `float iter_affine(int n, float a, float b, float c);` calculant le n-ième terme de la suite u_n de la suite lorsque f est la fonction affine $f(x) = ax + b$.

Q. 8 Au moyen d’expérimentations conjecturer les hypothèses nécessaires sur `a` et `b` pour avoir convergence de la suite. Appuyer ces conjectures d’une représentation graphique (ou mieux, d’une preuve de vos conjectures).

Q. 9 Que se passe-t-il lorsque l’on choisit $\texttt{c} = 3$ et f la fonction $f(x) = x + \sin(x)$.

Exercice 3 : Autour de la suite de Syracuse

On appelle suite de Syracuse de départ x, la suite $(u_n)_{n\in\mathbb{N}}$ définie par :

$$u_0 = x$$
$$u_{n+1} = \begin{cases} \frac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{sinon} \end{cases}$$

Q. 1 Définir une fonction `int prochain_syracuse(int un);` calculant, à partir d’un terme `un` de la suite, le terme suivant dans la suite de Syracuse.

Q. 2 Donner une fonction `int nieme_terme(int x, int n);` prenant en argument un point de départ `x` et un entier `n` et calculant le `n`-ième terme de la suite de Syracuse de départ `x`.

Q. 3 Donner une fonction `int duree_vol(int x);` prenant en argument un point de départ `x` et calculant l’indice du premier terme de la suite atteignant 1.

Q. 4 Donner une fonction `int plus_long_vol(int n);` prenant en argument un entier `n` et donnant le plus long vol pour les points de départ de l’intervalle $[\![1; \texttt{n}]\!]$.

Exercice 4 : Un mini-jeu “C’est plus, c’est moins”

Q. 1 Générer un entier aléatoire $x \in [\![0; 1000[\![$.
Demandez ensuite un nombre entier à l'utilisateur :
- Si il est au dessus, dites le lui et redemandez ;
- Si il est en dessous, dites le lui et redemandez ;
- Si il trouve, c’est gagné !

Q. 2 Inverser le fonctionnement du jeu pour que cette fois le joueur choisisse le nombre à deviner, et que ce soit l’ordinateur qui devine.

Exercice 5 : Calcul de volumes

Dans cet exercice nous nous intéressons à l’estimation du volume d’une forme géométrique en 3 dimensions, à la manière de l’algorithme vu en classe calculant une valeur approchée de π par estimation de la surface d’un cercle. L’exemple choisi (qui pourra/devra être changé en fin d’exercice pour tester sur d’autres exemples) sera un cylindre d’axe la droite $x = y = 0.5$ de rayon 0.25 et allant de $z = 0$ à $z = 2$.

Q. 1 Définir une fonction `int mon_volume(float x, float y, float z);` renvoyant vrai si et seulement si le point de coordonnées $(\mathtt{x}, \mathtt{y}, \mathtt{z})$ se trouve à l’intérieur du volume considéré (on rappelle que, pour l’instant, c’est un cylindre).

Q. 2 Définir une fonction `int est_dedans(float x, float y, float z, float size);` retournant vrai si et seulement si le cube, dont le coin le plus proche de $(0, 0, 0)$ a les coordonnées $(\mathtt{x}, \mathtt{y}, \mathtt{z})$ et de longueur de coté `size`, est tel que ses 8 sommets sont dans le volume.

Q. 3 Définir une fonction `int est_dehors(float x, float y, float z, float size);` retournant vrai si et seulement si le cube, dont le coin le plus proche de $(0, 0, 0)$ a les coordonnées $(\mathtt{x}, \mathtt{y}, \mathtt{z})$ et de longueur de coté `size`, est tel que ses 8 sommets sont hors du volume.

Q. 4 En déduire une fonction `void volume (int xl, int yl, int zl, int size, float prec);` calculant le volume du volume considéré dans le cube de recherche. Le cube de recherche est défini par son coin le plus proche de $(0, 0, 0)$: `xl`, `yl`, `zl` et par la longueur d’un de ses cotés : `size`. `prec` indique la précision attendue sur le volume. L’algorithme est le suivant :
- On découpe le cube de recherche en n^3 cubes (en découpant chaque dimension en n).
- On inspecte chacun des n^3 cubes pour savoir si :
  - Le cube est entièrement dans le volume (on approximera en utilisant les 8 sommets et la fonction `est_dedans`) ;
  - Le cube est entièrement hors du volume (on approximera en utilisant les 8 sommets et la fonction `est_dehors`) ;
  - Le cube n’est ni entièrement dans ni entièrement hors du volume.
- On obtient alors une sous-approximation du volume et une sur-approximation ;
- Si la différence entre la sur et la sous-approximation est supérieure à `prec`, on recommence en choisissant une valeur plus grande pour n
- Sinon on retourne la sous-approximation.

Q. 5 Vérifier votre fonction `volume` en utilisant vos connaissances mathématiques du volume d’un cylindre.