Task Sheet 1 - Simulating High-Dimensional Step Computations

In this task sheet, we explore the statistical properties of **random walks**.

Deliverables:

- A set of plots (saved to file)
- Code
- Simple explanation in point formats (For comparison and describing why it happens)

The submission should be bundled in a single zip file. The file should be named `yourname_sos_ex1.zip`. Please check the LEA course page for your submission deadline.

1 Exercise 1 - The Drunkard's Walk

A drunkard starts to walk from position $(x, y) = (0, 0)$ in 2D space and takes N steps. At each timestep, the drunkard turns in a random direction and takes a step of length $L = 1$.

a) Implement the drunkard's walk. You are free to use a programming language of your choice, but if you don't use `C`, `MATLAB`, `Python` or `C++`, please include a guide how to execute your code.

   Plot an example run of $N = 10^5$ steps. Describe the differences if you plot only the first 10^3 and 10^4 steps.

b) Now we put the drunkard inside a bounded circular environment. The random walk starts at the center of the environment. Implement the following scenarios and execute 10^4 independent runs of the random walk per case. Before each run, the drunkard's position is reset to $(0, 0)$. Each run should consist of 10^4 steps.

   Plot the distribution of endpoints of the walk using a 2D-histogram and describe the differences between the scenarios. To create a histogram, separate the range of occurring values into uniformly spaced intervals. Then, count and plot the number of occurrences per interval.

   i) The boundary is a **cliff**: the random walk ends at the boundary.
   ii) The boundary is a **wall**: the drunkard may not cross the boundary.
   iii) The environment uses **periodic boundary conditions**: when drunkards cross the boundary, they appear at the opposite side of the environment.
   iv) There is **no boundary**.

v) The environment uses **one-sided periodic boundary conditions**: when drunkards cross the boundary at a positive x coordinate, they appear at the opposite side of the environment. The boundary at a negative x coordinate should act like a wall.

After running the scenarios with 10^4 steps, try different numbers of steps and environment sizes.

2 Exercise 2 - The Sieve of Eratosthenes in C and performance

a) Implement the Sieve of Eratosthenes in C and in Python or Matlab. Run the algorithm for $n = 10^9$ and print out the highest prime number below n you found.

b) Compare the execution time between the implementation in C and in Python or Matlab.

Hint: If you have issues with `seg fault` errors in C, you should look at the way you allocate memory for your arrays. Dynamic memory management using the commands `malloc` or `calloc` might help you.

The `-O3` compiler flag might help improve the performance of your C code.