Chapter 1: Straight lines and the R programming language

1. Answer the following questions to improve your geometric intuition of straight-line equations: *(Note: Points are given in the form (x, y), where x is the horizontal coordinate and y is the vertical coordinate.)*
   Answer the following questions:

   - What is the equation of the straight line with a slope of -1.5 and an intercept of 5? Does this line pass through the point (2, 2)?
   - What is the equation of the straight line that passes through the points (0, 7) and (1, 9)?
   - What is the equation of the straight line that passes through the points (-3, 4) and (2, 4)?
   - What is the equation of the horizontal line (i.e., parallel to the x-axis) that passes through the point (3, 6)?
   - What is the equation of the straight line with a slope of 2.5 that passes through the point (3, 4)?
   - What is the equation of the straight line with an intercept of 1 that also passes through the point (10, 7)?
2. Open R and answer the following calculations using the R prompt:

   - $4829 + 5029 =$
   - $28 \times 82 =$
   - $5/3 =$
   - $7^5 =$
   - $\sqrt{20} =$ (hint: use the `sqrt()` function)
3. Using the R prompt, create and assign the following sequences to variables:

   - A sequence of increasing values from 0 to 100 in steps of 5
   - A sequence of decreasing values from 1 to 0 in steps of -0.05
4. Write a script that combines a `for` loop and an `if` statements to process numbers from 1 to 100 as follows:

   - Use a `for` loop to iterate over numbers from 1 to 100.
   - Inside the loop, check whether each number is *divisible by 2*.
     - *Hint:* Use the modulo operator `%%`, which returns the remainder of a division. A number is divisible by 2 if the remainder is `0`. Examples:
       - `4 %% 2` returns `0` (4 is divisible by 2)
       - `5 %% 2` returns `1` (5 is not not divisible by 2)
   - *If* a number is divisible by 2, print it to the console. Otherwise, move to the next number.
5. Write an R script to plot a straight line using the equation $y = mx + c$ where the slope $m = 2$ and the intercept $c = 1$. Define x as a sequence of numbers from -5 to 5, and then plot the resulting y values.

6. Indexing allows you to access or exclude specific elements of a vector. Create this vector:

```
x <- c(10, 20, 30, 40, 50, 65, 80, 33)
```

and answer the following questions:

- What is the value of `x[3]`?
- What values do you get with `x[c(2, 4)]`?
- What does `x[-1]` return? Does this command modify `x`?
- What happens if you try to access `x[10]`? Why?
- What values do you get with `x[x<35]`?
- Try assigning `x <- x[-2]`. What happens to the original vector?
- Now try `x[c(TRUE, FALSE, TRUE, FALSE)]`. What do you notice?

Hint: Think about how logical indexing works. What happens if the logical vector is shorter than the length of `x`?

7. Create a function in R to compute the distance travelled by an object moving with a constant speed of 5 metres per second. Use the formula $d = 5t$, where d is the distance and t is the time. Plot a graph of distance as a function of time for t values ranging from 0 to 10 seconds.

8. Write an R script that generates two lines with random slopes and intercepts using the command `rnorm(1)` (e.g., `a <- rnorm(1)` to assign a random value to `a`). Next, calculate which line has the steeper slope. Plot both lines on the same graph and use visual cues (such as line thickness or colour) to highlight the steeper line. The script should function each time it is executed (sourced), meaning that new lines with random slopes and intercepts are generated every time the script runs.
Hint: Use an `if` statement to check which line has the steeper slope and ensure that the line with the higher slope is always drawn thicker or in a more salient colour each time the script runs.