The following exercises should be completed in the Racket programming language [1]. Remember to plan your work and make regular commits to your repository. The instructions for submitting your work are given on the Moodle page.

1. Write, from scratch, a function in Racket that uses a brute-force algorithm that takes a single positive integer and return true if the number is a prime and false otherwise. Call the function `decide-prime`.

2. Write, from scratch, a function in Racket that takes a positive integer n_0 as input and returns a list by recursively applying the following operation, starting with the input number.

$$n_{i+1} = \begin{cases} 3n_i + 1 & \text{if } n_i \text{ is odd} \\ n_i \div 2 & \text{otherwise} \end{cases}$$

   End the recursion when (or if) the number becomes 1. Call the function `collatz-list`. So, `collatz-list` should return a list whose first element is n_0, the second element is n_1, and so on. For example:

```
> (collatz-list 5)
'(5 16 8 4 2 1)
> (collatz-list 9)
'(9 28 14 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1)
> (collatz-list 2)
'(2 1)
```

3. Write a function `hamming-weight` in Racket that takes a list l as input and returns the number of non-zero elements in it. For example:

```
> (hamming-weight (list 1 0 1 0 1 1 1 0))
5
```

4. Write a function `hamming-distance` in Racket that takes two lists and returns the number of positions in which they differ. For example:

```
> (hamming-distance (list 1 0 1 0 1 1 1 0) (list 1 1 1 1 0 0 0 0))
5
```

References

[1] PLT Inc. Racket – a programmable programming language.