The following exercises are related to the Racket programming language [3].

1. Re-write the following expressions in Racket and evaluate them using a Racket interpreter/compiler.
   (a) $(3 \times (5 + (10 \div 5)))$
   (b) $(2 + 3 + 4 + 5)$
   (c) $(1 + (5 + (2 + (10 \div 3))))$
   (d) $(1 + (5 + (2 + (10 \div 3.0))))$
   (e) $(3 + 5) \times (10 \div 2)$
   (f) $(3 + 5) \times (10 \div 2) + (1 + (5 + (2 + (10 \div 3))))$

2. Define a procedure `discount` that takes two arguments: an item's initial price and a percentage discount [2]. It should return the new price:

```
> (discount 10 5)
9.50
> (discount 29.90 50)
14.95
```

3. Write a function called `appearances` that returns the number of times its first argument appears as a member of its second argument [2].

4. Write a procedure `inter` that takes two lists as arguments. It should return a list containing every element that appears in both lists, exactly once.

5. Write a procedure `noatoms` that takes a list and returns the number of atoms it contains.

6. Here is a Racket procedure that never finishes its job:

```
(define (forever n)
  (if (= n 0)
      1
      (+ 1 (forever n))))
```

   Explain why it doesn't give any result[2].

7. Write a function called `range` that takes an integer n and returns a list containing the atoms 1, 2, 3, ..., n.

8. Write a function called `reversel` that takes a list and returns it reversed.

9. If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23. Write a procedure to find the sum of all the multiples of 3 or 5 below 1000 [1].

10. Write a procedure called `flatten` that takes as its argument a list, possibly including sublists, but whose ultimate building blocks are atoms. It should return a sentence containing all the atoms of the list, in the order in which they appear in the original:

```
> (flatten '(((a b) c (d e)) (f g) ((((h))) (i j) k)))
(a b c d e f g h i j k)
```

11. Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:

$$1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$$

By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms [1].

12. Write a procedure `to-binary`:

```
> (to-binary 9)
1001
> (to-binary 23)
10111
```

13. Write Heap's algorithm for generating permutations in Racket.

References

[1] Project Euler. Project euler.

[2] Brian Harvey and Matt Wright. Simply scheme: Introducing computer science.

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