An R programmer can determine the order of processing of commands, via use of the control statements; `repeat{}` , `while()` , `for()` , `break` , and `next`

Exercise 1

The `repeat{}` loop processes a block of code until the condition specified by the `break` statement, (that is mandatory within the `repeat{}` loop), is met.

The structure of a `repeat{}` loop is:

```
repeat {
commands
if(condition) {
break
}
}
```

For the first exercise, write a `repeat{}` loop that prints all the even numbers from `2` – `10` , via incrementing the variable, " `i <- 0` ".

Exercise 2

Using the following variables:

```
msg <- c("Hello")
i <- 1
```

Write a `repeat{}` loop that breaks off the incrementation of, " `i` ", after `5` loops, and prints " `msg` " at every increment.

Exercise 3

`while()` loop will repeat a group of commands until the condition ceases to apply. The structure of a `while()` loop is:

```
while(condition) {
commands
}
```

With, `i <- 1` , write a `while()` loop that prints the odd numbers from `1` through `7` .

Exercise 4

Using the following variables:

```
msg <- c("Hello")
i <- 1
```

Write a `while()` loop that increments the variable, " `i` ", `6` times, and prints " `msg` " at every iteration.

Exercise 5

The `for()` loop repeats commands until the specified length of the condition is met. The structure of a `for()` loop is:
`for(condition) { commands }`

For example:

```
for(i in 1:4) {
print("variable"[i])
}
```

```
for(i in seq("variable")) {
print(i)
}
```

```
for(i in seq_along("variable")) {
print("variable"[i])
}
```

```
for(letter in "variable") {
print(letter)
}
```

For this exercise, write a `for()` loop that prints the first four numbers of this sequence: `x <- c(7, 4, 3, 8, 9, 25)`

Exercise 6

For the next exercise, write a `for()` loop that prints all the letters in `y <- c("q", "w", "e", "r", "z", "c").`

Exercise 7

The `break` statement is used within loops to exit from the loop. If the `break` statement is within a nested loop, the inner loop is exited, and the outer loop is resumed.

Using `i <- 1` , write a `while()` loop that prints the variable, " `i` ", (that is incremented from `1` – `5`), and uses `break` to exit the loop if ' `i` " equals `3` .

Exercise 8

Write a nested loop, where the outer `for()` loop increments " `a` " `3` times, and the inner `for()` loop increments " `b` " `3` times. The `break` statement exits the inner `for()` loop after `2` incrementations. The nested loop prints the values of variables, ' `a` " and " `b` ".

Exercise 9

The `next` statement is used within loops in order to skip the current evaluation, and instead proceed to the next evaluation.

Therefore, write a `while()` loop that prints the variable, " `i` ", that is incremented from `2` – `5` , and uses the `next` statement, to skip the printing of the number `3` .

Exercise 10

Finally, write a `for()` loop that uses next to print all values except ' `3` " in the following variable: `i <- 1:5`