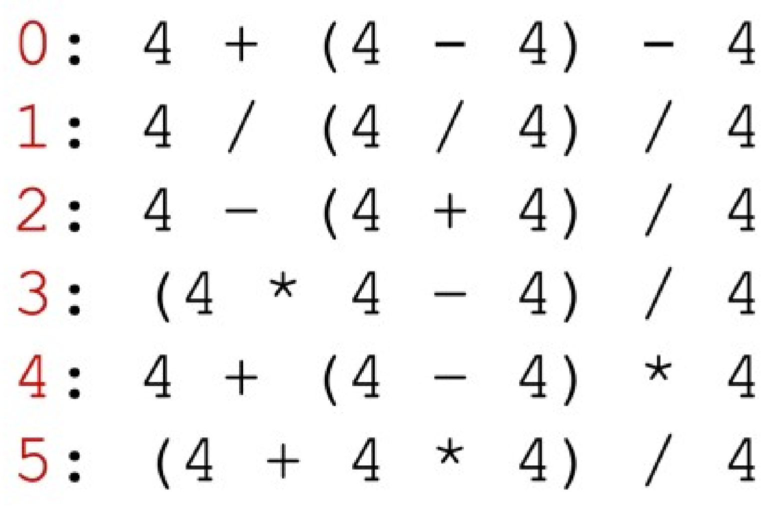
Yes. Playing with numbers is another call for useless R function. This time, we will be using a function that will find the simplest mathematical expression for a whole number from 0 to 9 (or even higher), using only:
– common mathematical operations (and symbols)
– only four digits of four (hence name Four Fours)
– no concatenations of the numbers (yet).



```
0: 4 + (4 - 4) - 4
1: 4 / (4 / 4) / 4
2: 4 - (4 + 4) / 4
3: (4 * 4 - 4) / 4
4: 4 + (4 - 4) * 4
5: (4 + 4 * 4) / 4
```

For the sake of brevity, I have only used couple of numbers, but list can go on and on. And also only four mathematical operations are used:
– addition
– subtraction
– multiplication
– division.

I next version we can also add:
– exponentiation
– factorial
– root extraction.

To make calculations a little bit faster to compute, forcing order of operations by adding parentheses is also a helpful way to do it. Next version should have multiple-parentheses available.

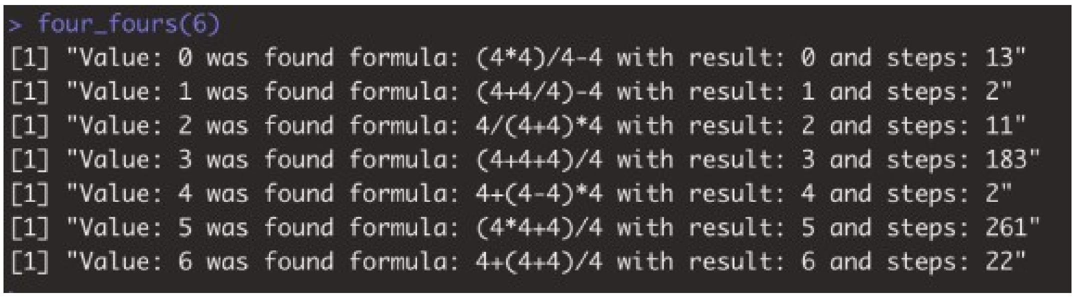
Useless function:

```
# Four Fours function
four_fours <- function(maxnum) {
  for (i in 0:maxnum) {
      oper <- c("+","*", "-", "/")
      para <- c("(",")")
      step_counter <- 0
      res <- i + 1
        while (i != res) {
        oper3 <- sample(oper,4,replace=TRUE)
        for44 <- paste0("4",oper3[1],"4",oper3[2],"4",oper3[3],"4")

        #adding paranthesis
        stopit <- FALSE
        while (!stopit){
          pos_par <<- sort(sample(1:7,2))
          nn <- pos_par[1]
          mm <- pos_par[2]
          rr <<- abs(nn-mm)

          if (rr == 4 | rr == 5 ){
            stopit <- TRUE

          }
        }

        for44 <- paste0(substr(for44, 1, nn-1), "(", substr(for44, nn,
nchar(for44)), sep = "")
        for44 <- paste0(substr(for44, 1, mm-1+1), ")", substr(for44, mm+1,
nchar(for44)), sep = "")

       # if (for44 ) like "(/" or "(-" or "(*" or "(+" -> switch to -> "/(" or
"-("
        for44 <- gsub("\\(-", "-\\(", for44)
        for44 <- gsub("\\(/", "/\\(", for44)
        for44 <- gsub("(*", "*(", for44, fixed=TRUE)
        for44 <- gsub("(+", "+(", for44, fixed=TRUE)
        for44 <- gsub("\\+)", "\\)+", for44)
        for44 <- gsub("\\-)", "\\)-", for44)
        for44 <- gsub("\\*)", "\\)*", for44)
        for44 <- gsub("\\/)", "\\)/", for44)

        ### Adding SQRT
        if (i >= 10){
          lii <- lapply(strsplit(as.character(for44), ""), function(x) which(x
== "4"))
          start_pos <- sample(lii[[1]],1)
          for44 <- paste0(substr(for44, 1, start_pos-1), "sqrt(",
substr(for44, start_pos, start_pos), ")", substr(for44, start_pos+1,
nchar(for44)),sep = "")
        }

        ###  Adding Factorial
        if (i >= 11){
          li <- lapply(strsplit(as.character(for44), ""), function(x) which(x
== "4"))
          start_pos_2 <- sample(li[[1]],1)
          for44 <- paste0(substr(for44, 1, start_pos_2-1), "factorial(",
substr(for44, start_pos_2, start_pos_2), ")", substr(for44, start_pos_2+1,
nchar(for44)),sep = "")
        }

        res <- eval(parse(text=for44))
        step_counter <- step_counter + 1
        if (res==i){
          print(paste0("Value: ", res, " was found formula: ", for44, " with
result: ", res, " and steps: ", step_counter, collapse=NULL))
        }
        }
      i <- i + 1
    }
}
```

And to run the function, simple as:

```
#Run function
four_fours(6)
```

After the finish run, results are displayed with number of tries and the formula.



```
> four_fours(6)
[1] "Value: 0 was found formula: (4*4)/4-4 with result: 0 and steps: 13"
[1] "Value: 1 was found formula: (4+4/4)-4 with result: 1 and steps: 2"
[1] "Value: 2 was found formula: 4/(4+4)*4 with result: 2 and steps: 11"
[1] "Value: 3 was found formula: (4+4+4)/4 with result: 3 and steps: 183"
[1] "Value: 4 was found formula: 4+(4-4)*4 with result: 4 and steps: 2"
[1] "Value: 5 was found formula: (4*4+4)/4 with result: 5 and steps: 261"
[1] "Value: 6 was found formula: 4+(4+4)/4 with result: 6 and steps: 22"
```

There are some solutions that do not need factorial or root extraction and some, that do. Therefore from step 10 till 14, there are some need for either and from 15 till 18, simple operations are enough, and so on. At some integer, also concatenation of two fours would be required.

The above solution should easily generate solutions from 1 to 25, based on my tests. There are also many optimisations possible; one useless would be using cloud scalable architecture – most useless (and expensive, though) optimisation – which I certainly do not approve of.