R tasks :

Please, create a function for each task. Functions should be named as task names, usually, as `task1`, `task2`, etc. Here are some examples of functions:

```
fun1 <- function(x, y) { # here are arguments in the brackets
  x + y # the result of the function is its last expression
}

# let's now call the function
fun1(10, 20) # this is 30

# another example
fun2 <- function(n) {
   c(1:n, n:1)
}

# let's test the function
fun2(5) # this is a vector 1, 2, 3, 4, 5, 5, 4, 3, 2, 1

fun3 <- function() {
    x <- rbinom(20, size=1, prob=0.5)  # toss a coin 20 times
    mean(x) # this is the result of a function, because this is the last
expression
}

fun3() # the result is always different, but it should be about 0.5
```

1. **task1** Read help about the `rep` function (type `help(rep)`). Then use it to produce a vector of repeating numbers 1, 2, and 3: 1,2,3,1,2,3,1,2,3,…

- . The length of the vector should be 40.

  - **task1a** You are given integers `n` and `size`, create a vector of repeating numbers from `1` to `n` of size `size`, for example, `task1a(3, 10)` should return `c(1, 2, 3, 1, 2, 3, 1, 2, 3, 1)`.

- **task2** You are given integers `n` and `size`. Using the function `sample` (read help if needed) generate a vector of size `size` consisting of random integers from `1` to `n`, and return it. For example, `task2(3, 5)` may return `1, 3, 2, 3, 3`.

  - **task2a** call task2a with the arguments n=5 and size=100, then use functions `table` to find out, how many times each number was generated, use print to print the table from inside the function.

- **filter.k** You are given a vector `x` and a number `k`. Return a new vector, that is a copy of `x` but without elements equal to `k`. For example, `filter.k(c(1, 2, 3, 4, 3, 2, 1), 3)` should return `1, 2, 4, 2, 1`.
- **random.walk.1d** You are given an integer `steps`, generate a random vector of size `steps` consisting of numbers 1 and -1. Return the sum of its elements.

- **mixed.distribution**. You are given an integer `size`. Generate a vector of size `size` using the following algorithm: to generate the next number, toss a coin (50%/50% of head and tails). If you get heads, generate a number from distribution $N(0,1)$

5. . Otherwise, from a uniform distribution with `min=-1` and `max=1`.
6. **random.walk.2d** Random walk on a plane. You are given an integer `steps`. Generate `steps` times a pair of numbers, this pair may be either (1 0), (-1 0), (0 1), or (0 -1). Put all this pairs in one vector, so you get a vector of size `2 * size`. This pairs correspond to movements of a point on a plane, a pair is a change for x and y coordinates correspondingly. That is a point moves either up, left, right or down. Finally, sum all x coordinates (all odd indexes), then sum all y coordinates (all even indexes), and you will get two coordinates of where a point had come after its random walk.
    - **plot.random.walk** You are given interers `n` and `size`. Call the previous function with the `size` argument `n` times. Plot all the `n` points.
7. **kinder.surprise** There are `n` possible toys inside a kinder surprise. How many kinder surprises should one buy in avarage to get all the toys? Make 10000 experiments, where you repeat opening kinder surprises untill you get all the toys. Then compute an avarage number of steps you needed to finish.

R datasets:

1. Create a script (not a function) `task1.R` with the following computations:
    1. Save dataframe `Cars93` from the library `MASS` to the variable `cars`.
    2. Add the column `kpl` (kilometers per liter) to the dataframe, use a column `MPG.city` (miles per gallon)
    3. Select all cars with `Horsepower` greater than 200.
    4. Plot Horsepower vs kpl. Make informative labels for axes.
        - add a linear regression line.
    5. Make a copy of the `cars` dataframe, that has only columns `Horsepower`, `kpl`, and `wt` (weight in tonns, use the original `Weight` column that has weights in pounds). Pass this copy as the only argument for the `plot` function. What do you see on this plot?
2. Create a script (not a function) `task2.R`.
    1. Create a new dataframe about balls, you will use random values for it. It should have two columns`size` and `color`. The first one is a factor with levels “big” and “small”, the second one is a factor with three levels “red”, “green”, “blue”. Use `sample` to create a create a vector of 1000 random values.
    2. Apply the following functions to this dataframe: `table`, `summary`, `plot`. How many big green balls are there?
    3. Add the column `weight` this should be normally distributed random numbers with different means for different balls sizes.
    4. Call the functions `table`, `summary`, `plot` again. What do they show?
    5. Try to make a plot of weight vs. size, but points on the plot should have the same color as the corresponding ball.