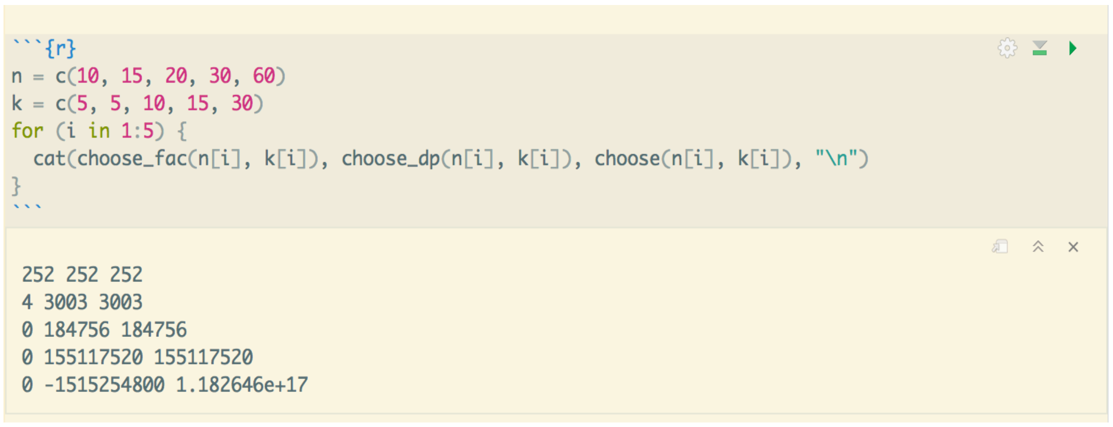

~~~r
```{r}
n = c(10, 15, 20, 30, 60)
k = c(5, 5, 10, 15, 30)
for (i in 1:5) {
  cat(choose_fac(n[i], k[i]), choose_dp(n[i], k[i]), choose(n[i], k[i]), "\n")
}
~~~
```

```
252 252 252
4 3003 3003
0 184756 184756
0 155117520 155117520
0 -1515254800 1.182646e+17
```

The left, middle, right columns are the results calculated by choose_fac(), choose_dp(), choose() respectively. The right column, which is from choose(), is correct, so we can see the choose_dp() has 4 correct results, and choose_fac() has 1 correct result.

The reason for choose_fac()'s errors is that the numeric type of the function fac() is integer (up to 2147483647), so fac() would be inaccurate when its argument bigger than 12, because factorial(13) = 6227020800 > 2147483647. Thus, choose_fac() loses its functionality since the second round which has n being 15.

The innate reason for the error of the last calculation of choose_dp() is also for the limitation of numeric type. Since its calculation doesn't include factorials, and use addition instead, its overflow occurs much later. The biggest value during the calculation of choose_dp() is the final result we have, so we can see choosing 15 from 30 is 155117520, which is still in the pool. However, choosing 30 from 60 is much bigger than the integer max, so the overflow happens already in the recursion. Thus, the result is far from correction.
```