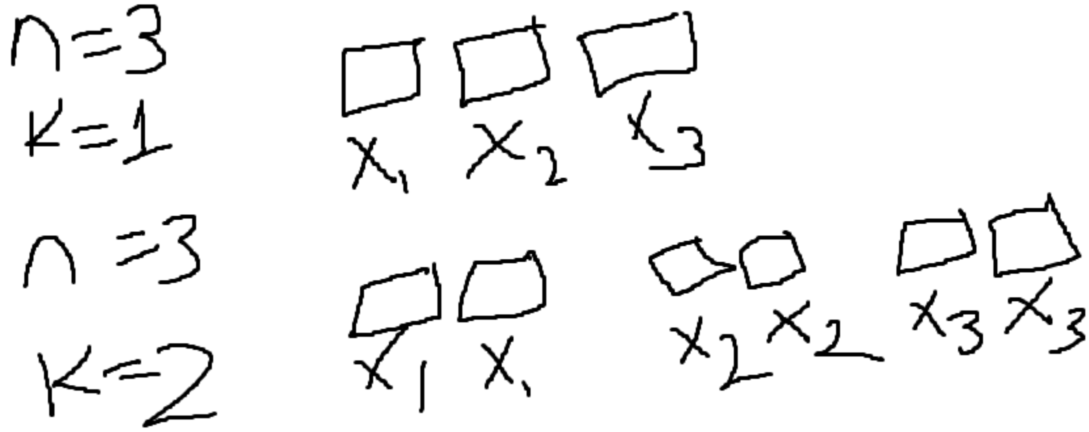
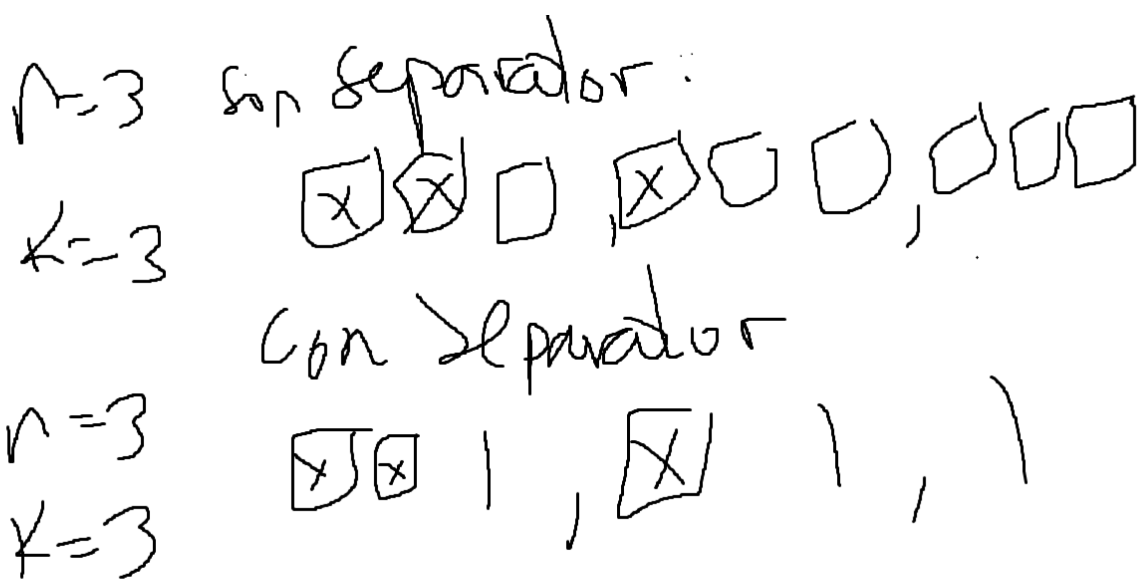
20.

Esta formula proviene del hecho de que se puede extrapolar el problema de repetición a un problema de una combinación sin repetición, aumentando el n sobre el cual se distribuye el grupo. Precisamente, la manera en que aumenta el número de los cuales extraemos los r grupos, es n+r-1. La forma en la que pensemos el problema tiene que ser tal que nos permita contar grupos dobles de una misma variable. La manera más fácil de hacer esto es “crear” otras n*k variables, tal que, si alguna de las nuevas entradas se escoge, se refieran a la repetición de una variable en específico.



El problema de esto es que la formula de combinación de este sistema es que le está dando la misma importancia del segundo x1, cuando esta variable solo interesa cuando se repite una misma variable. Para esto, se puede crear un objeto tal que haga referencia a la separación de dos variables, tal que no escatime por cuántas nuevas variables separe, sino por **las variables** que separen en sí. De esta manera eliminaríamos las variables extras que no se usen. Por ejemplo:



Los separadores tienen la misma importancia que las variables normales, por lo que en el conteo no debería de haber diferencia entre variable y separación. Nuestro siguiente objetivo sería ver cuántos separadores debería de haber. Como los separadores dependen de las variables que separen, deben haber n-1 (ya que solo "están en las mitades"). De esta manera tenemos que la cantidad de variables de este nuevo set es de n-1 barras + k variables escogidas. De estas, es que haremos un grupo de k sin repetición, que no es más que

$$\binom{n+k-1}{k} = \frac{(n+k-1)!}{k!(n-1)!}.$$

22.

Fijamos los 10 números y revisamos los números que faltan para que sumen hasta 10.

$$11 \left[\begin{array}{lll} 0 - & 10 - & 0 \\ & 9 - & 1 \\ & 8 - & 2 \\ & 7 - & 3 \\ & \ddots & \end{array}\right. + 10 \left[\begin{array}{lll} 1 - & 9 - & 0 \\ & 8 - & 1 \\ & 7 - & 2 \\ & \vdots & \end{array}\right. + 9 \cdots$$

$$= 11 + 10 + 9 \ldots + 1 = \frac{11(12)}{2} = 66$$

La respuesta es 66, y computacionalmente también nos da ese mismo numero:

```
In [7]: resp = []
        for i in range(0, 11):
            for k in range(0, 11):
                for z in range(0, 11):
                    if (i + k + z) == 10:
                        resp.append((i,k,z))
        print(len(resp))

        66
```