Constructing Subsets of Size k from a Set of Size n

The goal of this assignment is to write the **R** function `k.Subsets.R` that constructs a matrix M that contains all subsets of k elements from a set of n elements.

Inputs:

1. k - The cardinality of each subset.
2. n - The cardinality of the set.

Output: M - the matrix that contains all the subsets.

Here is pseudocode of the function taken from *Combinatorial Algorithms*, by Reingold, Nievergelt, and Leo:

$c_0 \leftarrow -1$

for $i = 1$ **to** k **do** $c_i \leftarrow i$

$j \leftarrow 1$

while $j \neq 0$ **do** {
- output $(c_1, c_2, \ldots, c_k)$
- $j \leftarrow k$
- **while** $c_j = n - k + j$ **do** $j \leftarrow j - 1$
- $c_j \leftarrow c_j + 1$
- **for** $i = j + 1$ **to** k **do** $c_i \leftarrow c_{i-1} + 1$

Algorithm 5.8 Generation of combinations (k subsets) in lexicographic order.

There are several issues in translating this Algorithm 5.8 to **R**:

- The code above refers to c as a vector. So, when it says c_1, c_2, c_3, etc, we code this using **R**'s syntax $c[1]$, $c[2]$, $c[3]$, and so on.
- The code says to "output" the vector c. We don't want to do that. Instead, we want to build a matrix M that contains all the subsets of length k. So, where it says to output the vector c, we will update a row of M.
- Speaking of M, we will have to initialize it as an $\binom{n}{k}$-by-k matrix full of zeros. We can do this using **R**'s `matrix` command. And, we will have to initialize an index – I called mine $M.row$ – that points to the first row of M. Then, every time a new subset, which is the vector c, is constructed, place the c in the row pointed to by $M.row$ and increment $M.row$.
- **R** indexes vectors beginning at 1. So, since the very first line of code above says to set c_0 to -1, we will have to work around that.

Here are several examples to try.

```
> k.Subsets(1,4)
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
> k.Subsets(2,4)
     [,1] [,2]
[1,]    1    2
[2,]    1    3
[3,]    1    4
[4,]    2    3
[5,]    2    4
[6,]    3    4
> k.Subsets(3,5)
      [,1] [,2] [,3]
 [1,]    1    2    3
 [2,]    1    2    4
 [3,]    1    2    5
 [4,]    1    3    4
 [5,]    1    3    5
 [6,]    1    4    5
 [7,]    2    3    4
 [8,]    2    3    5
 [9,]    2    4    5
[10,]    3    4    5
> k.Subsets(3,6)
      [,1] [,2] [,3]
 [1,]    1    2    3
 [2,]    1    2    4
 [3,]    1    2    5
 [4,]    1    2    6
 [5,]    1    3    4
 [6,]    1    3    5
 [7,]    1    3    6
 [8,]    1    4    5
 [9,]    1    4    6
[10,]    1    5    6
[11,]    2    3    4
[12,]    2    3    5
[13,]    2    3    6
[14,]    2    4    5
[15,]    2    4    6
[16,]    2    5    6
[17,]    3    4    5
[18,]    3    4    6
```

```
[19,]    3    5    6
[20,]    4    5    6
> k.Subsets(4,6)
      [,1] [,2] [,3] [,4]
 [1,]    1    2    3    4
 [2,]    1    2    3    5
 [3,]    1    2    3    6
 [4,]    1    2    4    5
 [5,]    1    2    4    6
 [6,]    1    2    5    6
 [7,]    1    3    4    5
 [8,]    1    3    4    6
 [9,]    1    3    5    6
[10,]    1    4    5    6
[11,]    2    3    4    5
[12,]    2    3    4    6
[13,]    2    3    5    6
[14,]    2    4    5    6
[15,]    3    4    5    6
```