Sorting is one of the most used operations in real life, where Computer Science comes into act. It is well-known that the lower bound of swap based sorting is nlog(n). It means that the best possible sorting algorithm will take at least O(nlog(n)) swaps to sort a set of n integers. However, to sort a particular array of n integers, you can always find a swapping sequence of at most $(n-1)$ swaps, once you know the position of each element in the sorted sequence.

For example consider four elements <1 2 3 4>. There are 24 possible permutations and for all elements you know the position in sorted sequence.

If the permutation is <2 1 4 3>, it will take minimum 2 swaps to make it sorted. If the sequence is <2 3 4 1>, at least 3 swaps are required. The sequence <4 2 3 1> requires only 1 and the sequence <1 2 3 4> requires none. In this way, we can find the permutations of N distinct integers which will take at least K swaps to be sorted.

Input

Each input consists of two positive integers N $(1 \leq N \leq 21)$ and K $(0 \leq K < N)$ in a single line. Input is terminated by two zeros. There can be at most 250 test cases.

Output

For each of the input, print in a line the number of permutations which will take at least K swaps.

Sample Input

```
3 1
3 0
3 2
0 0
```

Sample Output

```
3
1
2
```