## Problem J. Sequences

**Consider sequences of numbers. The first sequence consists of one number K.**
**Each subsequent sequence of numbers describes the previous one according to this rule.**
**We look through the described sequence from left to right and divide it into segments consisting of consecutive equal numbers (moreover, we always combine all consecutive identical numbers into one segment).**
**Further, each such segment is described by two numbers - the first number says how many times the same number is repeated, the second number says which number is repeated.**
**We write these pairs sequentially in accordance with the segments from left to right, and we get a new sequence (see examples below).**

**For example, for K = 2, the sequences will look like this:**

**No. Sequence How to read it (words in the description correspond to the numbers of the current sequence from left to right, and describe the previous sequence)**

**1 2 Original sequence**
**2 1 2 One "deuce"**
**3 1 1 1 2 One "one", one "two"**
**4 3 1 1 2 Three "ones", one "two"**
**5 1 3 2 1 1 2 One "three", two "ones", one "two"**
**6 1 1 1 3 1 2 2 1 1 2 One "one", one "three", one "two", two "ones", one "two"**

**Write a program that, given the original number K, will print the Nth resulting sequence.**

**Input data**
**Enter the number K (1 ≤ K ≤ 9) and the number N (1 ≤ N ≤ 15).**

**Output**
**Your program should print the Nth sequence derived from the initial sequence of one K. Separate numbers in the output with spaces.**

**Examples of input data**
**2->k**
**6**

**output**
**1 1 1 3 1 2 2 1 1 2**

**input data**
**2**
**one**

**output**
**2**

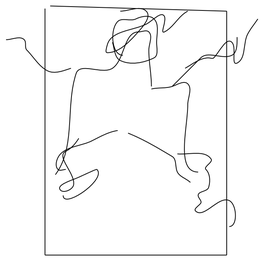
**input data**
**one**
**3**

**output**
**2 1**

3 3 3 3 3 1 1 1 1 2 0 - counter 10
0 0 1 1 1 1 1 2 0
5 3 1 1 1 1 2 0
5 3 4 1 1 2

Continue sequence
1,
11,
21, 1112
1112,
3112,
211213,
312213,
212223,
114213,
31121314,
41122314,
31221314 ...

idea1 There are no miracles in programming!
idea1 Programming like painting!



| server1 | server2 |
|---|---|
| php 5.2 | php 5.21 |
| programm1 | programm1 |

```
void sequences(int initialValue, int amountOfSteps) {
  int array1[1000] = {};
  array1[0] = initialValue;
  int array2[1000] = {};
  int counter = 1;            // has count of sequence in array
  int counter2 = 0;           //  counter for array2
  int buffer = -1;            // has memory of equal numbers in array
  int bufferCounter = 0;      // counts how many numbers equal to buffer
  while(amountOfSteps > 1) { // loop for sequence # on the left
     for(int i = 0; i < counter + 1; i++) {  // loop for sequenc eof numbers
        // std::cout << "array1[i] = "<< array1[i] << std::endl;
        // std::cout << "buffer = "<< buffer << std::endl;
        if (buffer == -1) {              // first number encountered
           buffer = array1[i];
           bufferCounter++;
        }
        else if (array1[i] == buffer) {   // first number encountered, remember and start counting
           bufferCounter++;
        }
        else if (array1[i] != buffer) {
           array2[counter2] = bufferCounter;
           counter2++;
           array2[counter2] = buffer;
           counter2++;
           buffer = array1[i];
           bufferCounter = 0;
           if (buffer == 0) {
              break;
           }
           bufferCounter++;
        }
           // std::cout << "buffer = "<< buffer << std::endl;
           // std::cout << "counter2 = "<< counter2 << std::endl;
           // printArray(array2, counter2);
     }
     // std::cout << "buffer = "<< buffer << std::endl;
     // std::cout << "counter2 = "<< counter2 << std::endl;
     // printArray(array2, counter2);

     counter = counter2;
     counter2 = 0;
     buffer = -1;
     // std::cout << counter << std::endl;
     for(int j = 0; j < 1000;j++ ) {
        array1[j] = array2[j];
        array2[j] = 0;
     }
     printArray(array1, counter);
     amountOfSteps--;
  }
}
```