DIFFICULTY:

TBD

PREREQUISITES:

Observation

PROBLEM:

Given integers N and K, construct a circular array of size N such that every subarray of size K contains distinct elements, while minimizing the maximum element.

EXPLANATION:

Obviously, we need at least K colors.

A natural-seeming starting point is to then attempt to repeat these K colors, to form $[1, 2, 3, \ldots, K, 1, 2, \ldots, K, 1, 2, \ldots]$.

This will allow you to form $\left\lfloor \frac{N}{K} \right\rfloor$ 'blocks' of $[1, 2, \ldots, K]$, and will leave the last few elements uncolored — specifically, the last $N\%K$ elements will be uncolored.

Using $N\%K$ more colors gives us a valid array, of course, and we've used $K + N\%K$ colors. Can we do any better?

Turns out, we can!
The idea is not too hard: as before, let's place $\left\lfloor \frac{N}{K} \right\rfloor$ blocks of $[1, 2, \ldots, K]$.
This leaves us with $N\%K$ uncolored elements. Instead of placing them all at the end of the array, let's instead distribute them equally (as much as possible) to the ends of all the blocks.

For example, if there are 21 blocks and 118 uncolored elements, give 6 elements each to the first 13 blocks and 5 elements each to the remaining ones.
In particular, each block will receive either $\left\lceil \frac{N\%K}{B} \right\rceil$ or $\left\lfloor \frac{N\%K}{B} \right\rfloor$ uncolored elements, where B is the number of blocks.

Now notice that the resulting array can be colored quite easily using $\left\lceil \frac{N\%K}{B} \right\rceil$ extra colors, giving us a solution using $K + \left\lceil \frac{N\%K}{B} \right\rceil$ colors.
Constructing this coloring is fairly straightforward: as noted above, each block of uncolored elements has either size y or $y - 1$, where $y = \left\lceil \frac{N\%K}{B} \right\rceil$. So,

- Color a block of size y with colors $K + 1, K + 2, \ldots, K + y$
- Color a block of size $y - 1$ with colors $K + 1, K + 2, \ldots, K + y - 1$

That is, the final coloring will look like $[1, 2, 3, \ldots, K, K + 1, \ldots, K + y, 1, 2, \ldots, K + y, 1, \ldots, K + y, \ldots, 1, 2, \ldots, K + y - 1, 1, 2, \ldots, K + y - 1]$

It turns out that this is also optimal.

▾ Proof

Suppose we are able to use C colors to color the array.
Let M be the maximum frequency of one of the colors; w.l.o.g say color 1.

Then, obviously, it must hold that $M \cdot C \geq N$.
Further, there are at least $K-1$ other elements between any two occurrences of 1.
This gives us a minimum of $M \cdot (K-1) + M = M \cdot K$ elements in the array, i.e, $M \cdot K \leq N$.

This tells us that $M \leq \lfloor \frac{N}{K} \rfloor$.

Now, recall that we have $M \cdot C \geq N$. We'd like to minimize C, so of course M should be as large as possible.
In other words, we choose $M = \lfloor \frac{N}{K} \rfloor$.

This tells us that

$$C \geq \left\lceil \frac{N}{\lfloor \frac{N}{K} \rfloor} \right\rceil$$

must hold.

The smallest C that satisfies this equation is indeed $K + \left\lceil \frac{N\%K}{\lfloor \frac{N}{K} \rfloor} \right\rceil$: this can be verified algebraically.

Of course, to solve the problem you don't need to solve this algebraically: you can, for example, run a loop on C or binary search to find the first time $C \cdot \lfloor \frac{N}{K} \rfloor \geq N$.

TIME COMPLEXITY

$\mathcal{O}(N)$ per test case.

```
for _ in range(int(input())):
    n, k = map(int, input().split())
    cols = k + (n%k + n//k - 1)//(n // k)
    print(cols)
    ans = [[i for i in range(1, k+1)] for _ in range(n//
k)]
    n -= k * (n // k)
    for col in range(k+1, cols+1):
        for list in ans:
            if n > 0:
                list.append(col)
                n -= 1
    for list in ans:
        print(*list, end = ' ')
    print()
```