COMPUTER SOCIETY OF INDIA (CSI)
COEP TECH STUDENT CHAPTER

PRESENTS

CODE QUEST 5.0

EDITIORIAL

THE GREAT DIVIDE

COEP

PROBLEM STATEMENT

At COEP Tech, the Data Structures and Algorithms (DSA) exam is notorious for pushing students to their intellectual limits. The stakes are high, and the pressure in the air is palpable. This year, the exam is scheduled to take place in the prestigious Cognizant Lab, a state-of-the-art facility equipped with rows of high-performance PCs. The exam is rumored to be especially tough this time, with whispers of complex recursive problems and elusive graph algorithms floating around the campus.

With the intensity of the challenge, some students might feel tempted to stray from the path of honesty and resort to "collaborative problem-solving" during the test. The vigilant professors have taken it upon themselves to prevent any such breach of integrity. They devised a cunning plan: ensure that no two students can sit close enough to even exchange glances, let alone whisper answers to each other.

PROBLEM STATEMENT

The Cognizant Lab has N + 2 PCs in a single row. The PCs at both ends are reserved for the invigilators, leaving N PCs in the middle for the students. These invigilators, like unyielding sentinels, ensure that the students feel the watchful gaze of authority.

When students enter the lab, they are assigned a PC following a strict set of rules :

1. Maximum Distance Rule : Each student tries to sit as far as possible from the nearest occupied PC.
2. Tie-Breaker Rule : If there are multiple PCs with the same minimum distance from the nearest occupied PC, the student chooses the one with the greatest maximum distance to an occupied PC.
3. Leftmost Rule : If there is still a tie, the student chooses the leftmost PC among the remaining options.

PROBLEM STATEMENT

As students file into the lab one by one, their choices are dictated by these rules, ensuring that the spacing remains optimal to deter any unfair practices.

The professor now wonders: after K students have taken their seats, what will the seating situation look like for the Kth student, the last one to enter the lab?
Specifically:
What is the maximum distance to the nearest occupied PC (maxD) for the Kth student?
What is the minimum distance to the nearest occupied PC (minD) for the Kth student?

The invigilators need this information to evaluate how effectively the seating arrangement minimizes opportunities for unfair means.

PROBLEM STATEMENT

For each empty PC (S), they compute two values, (LC) and (RC), where:
- LC is the number of empty PCs between (S) and the closest occupied PC to the left.
- RC is the number of empty PCs between (S) and the closest occupied PC to the right.

They then consider the set of PCs with the farthest closest neighbor, meaning the PCs for which min(LC, RC) is maximal. If there is only one such PC, they choose it. Otherwise, among those PCs, they select the one where max(LC, RC) is maximal.

Finally, the maxD is defined as max(LC, RC), and the minD is defined as min(LC, RC).

Your task is to solve for maxD and minD for all test cases!

PROBLEM STATEMENT

Suppose the Cognizant Lab setup is as follows:

N = 4 PCs available for students (with 6 PCs total, including the 2 reserved for invigilators).
K = 3 students entering.

Following the seating rules:

1. The first student takes the middlemost PC for maximum distance.
2. The second student takes the next farthest PC, ensuring maximum spacing.
3. The third student's seat is determined by the rules, and their maxD and minD values are calculated.

Your task is to solve for maxD and minD for all test cases!

INPUT FORMAT

The first line of the input gives the number of test cases, T. T lines follow. Each line describes a test case with two integers N (the number of available PCs for students) and K (the number of students entering the lab).

OUTPUT FORMAT & CONSTRAINTS

For each test case, output one line containing: `maxD minD`.

SOLUTION

```cpp
#include <bits/stdc++.h>
using namespace std;

map<long long, long long> ans;

long long cnt(long long n, long long m) {
    if (n < m)
        return 0;
    if (ans[n] != 0)
        return ans[n];
    long long a = (n + 1) / 2 - 1;
    long long b = n / 2;
    return ans[n] = cnt(a, m) + cnt(b, m) + 1;
}

int main() {
    int t;
    cin >> t;
    while (t--) {
        long long n, k;
        cin >> n >> k;

        long long l = 0, r = n;
        while (l < r) {
            long long mid = (l + r + 1) / 2;
            ans.clear();
            if (cnt(n, mid) < k)
                r = mid - 1;
            else
                l = mid;
        }

        cout << (l / 2) << ' ' << ((l + 1) / 2 - 1) << endl;
    }
    return 0;
}
```