Gandalf

Sauron and Saruman have been communicating from large distances using the Seeing Stones. Denethor, with great difficulty has been able to break into their channel of communication using his strength of will. Despite this however, it seems that their communication has been encrypted. Gandor's spies in Isengard have found out the encryption algorithm they use and have reported back to Denethor.

The algorithm is as follows:

We refer to a dequeue as a double-ended queue. We define a "dequeue permutation of N" as a permutation of 1 to N that can be got by starting from a dequeue having elements 1, 2, 3, . . . , N (in that order with 1 at the front and N at the back) and performing any sequence of N pop_front() or pop_back() operations.

Note that not all permutations of 1 to N are dequeue permutations. For example, with N = 3, you have 1-3-2-etc. as dequeue permutations whereas 2-1-3-1-2 aren't (you can't have 2 right at the beginning since its not at any end of the dequeue).

If Sauron wants to encrypt the number K and send it to Saruman, he would instead send the K-th lexicographically smallest (0-based indexed) dequeue permutation of N. That is, if Sauron wanted to send N to Saruman, he would just send 1-2-3- . . . -N (since this is clearly the smallest lexicographic dequeue permutation of N).

Sauron is transmitting the size of his army to Saruman, so that they can coordinate an attack on the domain of Rohan and Gondor. Since Sauron's will is so powerful, Denethor is able to get only vague glimpses of the numbers, and he is able to remember only the first half (floor(N/2)) elements of the permutation. Further, since those images are so vague, his understanding of the numbers happens out of order. For example, Denethor may understand that the 5th number of the permutation is 4, and later on only understand that the 3rd number was 3.

Help him estimate the minimum and maximum possible size of Sauron's army (value of K), given the number N, and incremental understanding of the first half of the permutation, not necessarily in order.

Input
For each testcase, output exactly floor(N/2) lines containing 2 integers each: j and i, denoting that Denethor has understood that the j-th element (1-based) of the supposed permutation is i.

Output
Between successive test cases, there should not be any blank lines in the output.

Constraints:
1 ≤ N ≤ 100,000
1 ≤ i ≤ floor(N/2)
1 ≤ j ≤ N

Notes/Explanation of Sample Input:

(a) the 1st number is 1.
(b) the 3rd number is 2 ... this means that the 2nd number has to be 32.
(c) the 2nd number is 32 ... this does not add any new information.
(d) the 4th number is 4. But this is not possible, since the 4th number can now either be 3 or 31.

Thus it is inconsistent (and none of the further observations can make it consistent). Also notice that in the 2nd test-case, the values have been output modulo 1,000,000,007.

Sample Input

1 2
2 3

Sample Output

1 2
2 3

Notes:

In the second test case, we see that:

(a) the 1st number is 1. 
(b) the 3rd number is 2 ... this means that the 2nd number has to be 32.  
(c) the 2nd number is 32 ... this does not add any new information. 
(d) the 4th number is 4. But this is not possible, since the 4th number can now either be 3 or 31. 

Hence it is inconsistent (and none of the further observations can make it consistent). Also notice that in the 2nd test-case, the values have been output modulo 1,000,000,007. 

Sample Input

2 3
0 73741863 536870912 805306367 536870912 805306367 
-1
-1 

Sample Output

2 3