6604 Airport Sort

There are few airlines that don't specify seat numbers before boarding. Instead, each passenger gets a ticket containing a unique integer in the range $[1, n]$ where n indicates the total number of seats in the airplane. Each ticket number belongs to a specified zone. Suppose a zone contains k tickets, then tickets with numbers $[1, k]$ are in zone 1; tickets with numbers $[k+1, 2k]$ are in zone 2 and so on. The last zone may contain less than k tickets if n is not divisible by k.

Before boarding, all the n passengers line up randomly in a straight line. In order to expedite the boarding process, it is convenient that the first k passengers in the line belong to zone 1, the next k in zone 2 and so on. There are two ways of rearranging the positions of the passengers.

1) Adjacent passengers keep swapping places until the required order is accomplished. Every second, only one pair of adjacent passengers can swap their places.

2) All the passengers walk simultaneously towards their correct positions. To walk from position x to position y, it takes $|x - y|$ seconds. $|x - y|$ is the absolute difference of x and y.

As you can guess, the first approach takes more time, but it's less 'noisy'. For this problem, you have to determine how much faster the 2nd approach is. More specifically, suppose the minimum time required to rearrange the passengers using the first approach is X and the minimum time for the second approach is Y, you have to find $X - Y$.

As an example, consider a plane of capacity 10 ($n = 10$) and zones of size 3 ($k = 3$). Suppose the passengers line up in the following order

```
3 7 1 2 4 6 5 8 10 9
```

Each integer represents the ticket number of the corresponding passenger. So, for the above example, the first passenger in the line has a ticket with number 3. Passengers with ticket numbers 7, 2, 5, 10 and 9 are not in their right positions.

Using the first strategy, it'd take a minimum of 6 swaps (and hence 6 seconds) to achieve the desired order.

I. swap 7 with 1 ⇒ `3 1 7 2 4 6 5 8 10 9`

II. swap 7 with 2 ⇒ `3 1 2 7 4 6 5 8 10 9`

III. swap 7 with 4 ⇒ `3 1 2 4 7 6 5 8 10 9`

IV. swap 7 with 6 ⇒ `3 1 2 4 6 7 5 8 10 9`

V. swap 7 with 5 ⇒ `3 1 2 4 6 5 7 8 10 9`

VI. swap 10 with 9 ⇒ `3 1 2 4 6 5 7 8 9 10` (now everyone is where they are supposed to be)

Using the second strategy, it'd take a minimum time of 5 seconds if we decide the final order to be (`3 2 1 4 6 5 7 8 9 10`). Here passengers with ticket 3, 1 and 8 are already in their right positions; it takes 1 second for passengers with tickets 4, 5, 6, 9 and 10 to get to their right positions; it takes 2 seconds for passenger with ticket 2 to come to her final position; it takes 5 seconds for passenger with ticket #7 to get to her right position. So this means, after 5 seconds, everyone will be at their right places using the 2nd strategy. There are other arrangements for which we get 5 seconds, but it's not possible to come up with something less.

So 2nd strategy is 1 second faster.

Input

The first line of input is an integer T ($T < 50$) that indicates the number of test cases. Each case consists of 2 lines. The first line contains 2 positive integers n ($n \leq 20000$) and k ($k \leq n$). The meanings of these variables are mentioned above. The next line contains n distinct integers in the range $[1, n]$. The first integer represents the ticket number of the passenger standing in front of the queue.

Output

For each case, first output the case number followed by the required result (i.e. how much faster the 2nd approach is).

Sample Input

```
3
10 3
3 7 1 2 4 6 5 8 10 9
11 3
1 2 3 4 5 6 7 8 9 10 11
5 2
5 4 3 2 1
```

Sample Output

```
Case 1: 1
Case 2: 0
Case 3: 4
```