Pagination

The **Dev Skill** team has a new task ahead. There is a rank list for the contest where the ranks of all contestants are shown in a list. But it will be a disaster to show hundreds of data in a single page and scroll down all the way down to see each data. To solve it, they need to add **pagination** on their rank list. Pagination is a way of organizing or numbering a lot of data within a webpage to make it more manageable and user friendly.

The rank list page contains pagination bar, where page numbers are shown. When a user click into a page number on the bar, the selected page data will show and pagination bar will reorganize for the selected page. Your task is to write a code for the various states of pagination bar according to selected current page.

For the pagination bar **Dev Skill** team is following below rules:

1. Pagination bar has specific format. Generally there will be two groups of page numbers separated by 5 dots. The first group must be shown. There will be no more than 6 numbers in first group and no more than 5 numbers in last group.
2. The last group will be shown if there are more page numbers after showing the first group. This last group will contain no more than 5 numbers including the last page number. If there are not enough pages after showing in the first group, the last group will be ignored along with the 5 dots.
3. You must show 6 pages in the first group if the total number of pages is greater than 5 otherwise shows as many page as there are in first group.
4. The first page number and the last page number must be shown always. Suppose there are 15 pages. Then page 1 and 15 must be shown in the bar. **1** 2 3 4 5 6 ….. 11 12 13 14 **15**.
5. If there are 6 pages in first group, then 4th position will have the current page number, 3 position will be current page number - 1, 2nd position will be current page number - 2, 5th position will be current page number + 1, 6th position will be current page number + 2 and 1st position will be always 1. For example, in this case: “1 4 5 **6** 7 8 ….. 11 12 13 14 15”, 6 is the current page number, so it is in 4th position. And accordingly 3rd position = 6-1 = 5, 2nd position = 6-2 = 4, 5th position = 6+1 = 7, 6th position = 6+2 = 8 and 1st position is 1.
6. The last group will have the last page number at the end, and decrease by 1 as long as the page number will not get duplicated with first group and maximum 5 numbers will be shown in the last group. For example, if the current page number is 6 and there are 15 pages in total, in this case the paging will be like this: “1 4 5 **6** 7 8 ….. 11 12 13 14 15”. Here 15 is the last page number as here is 15 pages in total, and each preceding positions in 2nd group decreased by 1 to create this 5 number group.
7. There should not be any duplicate page number in any of these groups.

Input

First line of input contains an integer (0 < T <= 50) denoting the number of test case. The first line of each test case contains the total page number (0 < TP <= 50) and next all lines contain integer which denoting the current page number (0 < CP <= TP). All number will be integer here.

Output

For each test case first print a line “Case X:” without double quote, where X denoting the current test case number. For each query of current page CP, print the output of current status of pagination tab where each number is separated by space. There are also a space between group of number and group of dot. There is no space between dots.

Sample Input	Sample Output
2 15 1 2 7 0 20 2 14 20 0	Case 1: 1 2 3 4 5 6 ..... 11 12 13 14 15 1 2 3 4 5 6 ..... 11 12 13 14 15 1 5 6 7 8 9 ..... 11 12 13 14 15 Case 2: 1 2 3 4 5 6 ..... 16 17 18 19 20 1 12 13 14 15 16 ..... 17 18 19 20 1 16 17 18 19 20

Limits:

Language	Time	Memory
C	1 Second	50MB
C++	1 Second	50MB
Java	1 Second	50MB
C#	1 Second	50MB
PHP	1 Second	50MB