E. Very Boring Homework

[Description]

Prof. Z thinks his homework is very hard for most of his students to solve (do you remember the task “Boring Homework”?) To his surprise, many students hand in correct solutions. He thinks the reason is actually the small size of the data set he used to test students’ programs rather than the low difficulty of the homework task. He decides to give his students the same homework again, but with enormous test cases. Of course, his students think his homework becomes even more boring this time. They need your help again.

For the ones who don’t know what homework Prof. Z. had given to his students last time:

You’re asked to draw a graph of a binary search tree (BST).

A binary search tree, which may sometimes also be called ordered or sorted binary tree, is a node-based binary tree data structure which has the following properties:

The left subtree of a node contains only nodes with keys less than the node's key.

The right subtree of a node contains only nodes with keys greater than the node's key.

Both the left and right subtrees must also be binary search trees.

--from Wikipedia

Given a list of integer keys that should be inserted into the BST one by one orderly, we can form a unique BST. Moreover, Prof. Z wants the students to draw the graph of this BST.

The rules to draw a graph of a BST are listed below:

1. The graph of a 1-node BST is a single 'o' (15th small Latin letter).
2. If a BST has a non-empty subtree, draw a single '|' just above the subtree's root, and a single '+' just above the previous drawn '|'. Finally, in the row of '+', use the least number (including 0) of '-'s to connect '+' (corresponding to the left or right subtree) and 'o' (denoting the parent node of the subtrees).
3. The left subtree (if exists) must be drawn on the left side of its parent. Similarly, the right subtree (if exists) must be drawn on the right side of its parent.
4. The column of the BST's root must not contain any characters belonging to left or right subtree.

5. For each node of the BST, the graph of its left subtree and the graph of its right subtree do not share common columns in the picture of the whole tree.

After the whole BST has been drawn, we number the rows from top to bottom, counting from 1, and so do the columns from left to right, counting from 1.

Due to the large scale of the tree, the graph will become so large that it is impossible for Prof. Z to check every detail of the graph this time. So you are only asked to hand in m fragments of that graph to Prof. Z instead of the whole one.

[Input]

The first line contains T, the number of test cases. T test cases follow.

For each test case:

The first line contains a positive integer N (N <= 100000).

The second line contains N distinct integers, each of which can be represented by a 32-bit signed integer. These numbers should be inserted into an empty BST one by one in the given order.

The third line contains an integer M (M <= 5).

M lines follow, each contains four integers, which are the row and column number of the top left corner, and the number of rows R_i and columns C_i of the required graph fragment, respectively. All the input integers will be positive and fit into a 32-bit signed integer, except R_i and C_i, which will be less than or equal to 200 and greater than 0.

[Output]

For each test case:

Output the case number counting from 1 in the first line.

Then M blocks follow, each contains R_i (or less, see next) lines. Each line should contain exactly C_i characters. Use space (ASCII 32) to fill in the blank. But if a line contains only whitespaces, this line should not be outputted.

Output a blank line after each graph fragment.

[Sample Input]

```
3
3
3 1 2
1
1 1 5 3
```

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

[Sample Output]

```
Case #1:
+-o
|
o+
 |
 o

Case #2:
+--o+
|   |
o-+ o+
  |  |
 +o  o
 |
 o

Case #3:
+o---
|
o  +-
   |
  +o+

o+
 |
 o-+
   |
  +o+
```