3356 Apple Tree

Figure 1 shows an apple tree in which each node has exactly one apple and every internal node has at least two children nodes.

Consider a worm in the apple tree. The worm visits apples (i.e., nodes) following the order of depth-first traversal from the root of the apple tree. (Notice that, the branching priority of an internal node is left-first order.) Figure 2 shows a binary representation of the depth-first traversal by a worm in the apple tree.

The binary representation is defined as follows: During the depth-first traversal,

'0' is printed, if a new node is visited, and

'1' is printed, if a node is returned after visiting all descendent nodes.

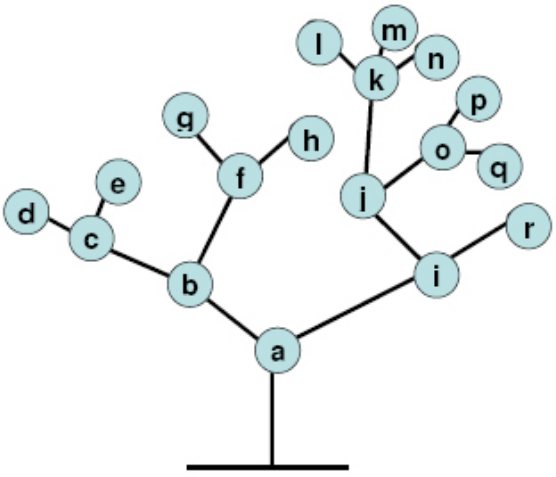

Figure 1

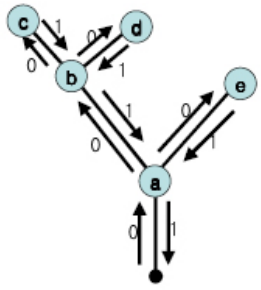

Figure 2

Binary representation	0	0	0	1	0	1	1	0	1	1
first visited node	a	b	c		d			e		
returned node				c		d	b		e	a

The second line of the table shows the first visited node corresponding to each '0' and the third line shows the returned node corresponding to each '1'. For example, the apple tree in Figure 1 can be represented by the definition above.

Binary representation	0	0	0	0	1	0	1	1	0	0	1	0	1	1	1	0	0	0	0	1	0	1	0	1	1	0	0	1	0	1	1	0	1	1	1
first visited node	a	b	c	d		e			f	g		h				i	j	k	l		m		n			o	p		q				r			
returned node					d		e	c			g		h	f	b					l		m		n	k			p		q	o	j		r	i	a

If there are wormy apples in the apple tree, we should remove them. When we cut a node `z`, every apple in the subtree rooted at node `z` (including the apple of node `z`) is going down. For example, apples `k`, `l`, `m`, and `n` are going down by cutting node `k` and apples `b`, `c`, `d`, `e`, `f`, `g` and `h` are going down by cutting node `b` in the apple tree in Figure 1. In order to remove wormy apples, we should find the subtree rooted at a node in which every wormy apple exists, and then cut the node.

Given two wormy apples (possibly same) in an apple tree, the problem is to find a node satisfying that two wormy apples must drop and the minimum number of normal apples drop simultaneously by

cutting the node. For example, suppose that nodes `c` and `f` are given as wormy apples in the apple tree in Figure 1. By cutting nodes `a` or `b` , two wormy apples can be removed. However, the smaller number of normal apples drop by cutting node `b` than the case that we cut node `a`. So, we want to find the node `b`.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case is composed of three lines. In first line, the number of nodes N ($3 \leq N \leq 2000$) is given. In next second line, $2N$ binary representation of an apple tree is given. There is no space between each representation. In last third line, two indices A and B ($1 \leq A, B \leq 2N$) corresponding to two wormy apples in the binary representation are given. These two indices are not corresponding to node directly but index of binary representation. Notice that, the value of each index can be '`0`' or '`1`'.

Output

Your program is to write to standard output. Print exactly one line for each test case. For each test case, find a node satisfying that the wormy apples must be removed and the minimum number of normal apples are removed simultaneously by cutting the node, and then print an index of '`0`' where the node was first visited and the other index of '`1`' where the node was returned in the binary representation. Print exactly one line for each test case index of '`0`' first, and index of '`1`' later. And there must be a single space between two indices.

Sample Input

```
3
18
000010110010111000010101100101110111
4 11
18
000010110010111000010101100101110111
11 12
18
000010110010111000010101100101110111
5 12
```

Sample Output

```
2 15
9 14
2 15
```