H. Rotating keyboard

Program:	`keyboard.(cpp|java)`
Input:	`keyboard.in`
Balloon Color:	`Green`

Description

You have a strange cellphone. The cellphone screen has two parts. The upper part shows the number that is typed. The lower part displays the digits and symbols and a cursor. However, the screen is not a touch screen. Therefore, the only way to type a digit is to attach a *detachable keyboard* to the cellphone screen, which consists of only four arrow keys and an OK button. The function of the arrow keys is only to move the cursor on the screen, without outputting anything. The function of the OK button is to output the digits on which the cursor is currently located. The initial position of the cursor is on the digit 5.

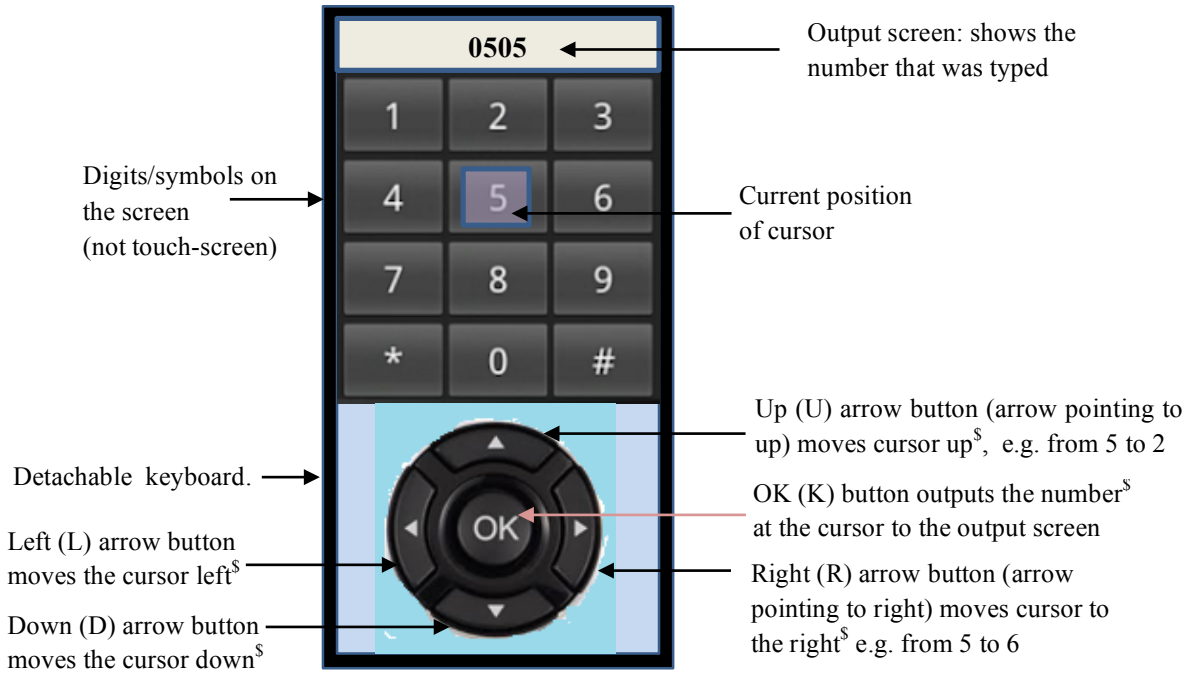

Figure H.1: The cellphone screens and keyboard. The shorthand names of the arrow keys will be L, R, U, and D corresponding to the arrow keys pointing to Left, Right, Up and Down. The OK key will be shorthanded as K. The number 0505 on the output screen was typed by the following key sequence (assuming initial cursor position on 5): DDKUUKDDKUUK.

$ see the terms and conditions next page.

So, for example, if someone wants to type the digit 0 (starting at the initial state, i.e., at button 5), he has to press the following key sequence: DDK. The first D will move the cursor down to 8, the second D will move it to 0, and then K will type 0. After typing the digit, the cursor location will remain on the digit 0. Given the initial position of the cursor at button 5, and a sequence of key-presses, you are to find the number (consisting of one or more digits) that was typed.

Wait, there is a catch! The detachable keyboard can be attached by rotating 90, 180, or 270 degrees clockwise. However, even after rotation the function of each arrow key remains the same, for example, the key that moved the cursor down before rotation, will still move the cursor down after

rotation. But according to the user (i.e., observer), the Left arrow key at the rotated position is different from the Left arrow key before rotation. As an example, if the keyboard is rotated 90 degrees, it will look as follows:

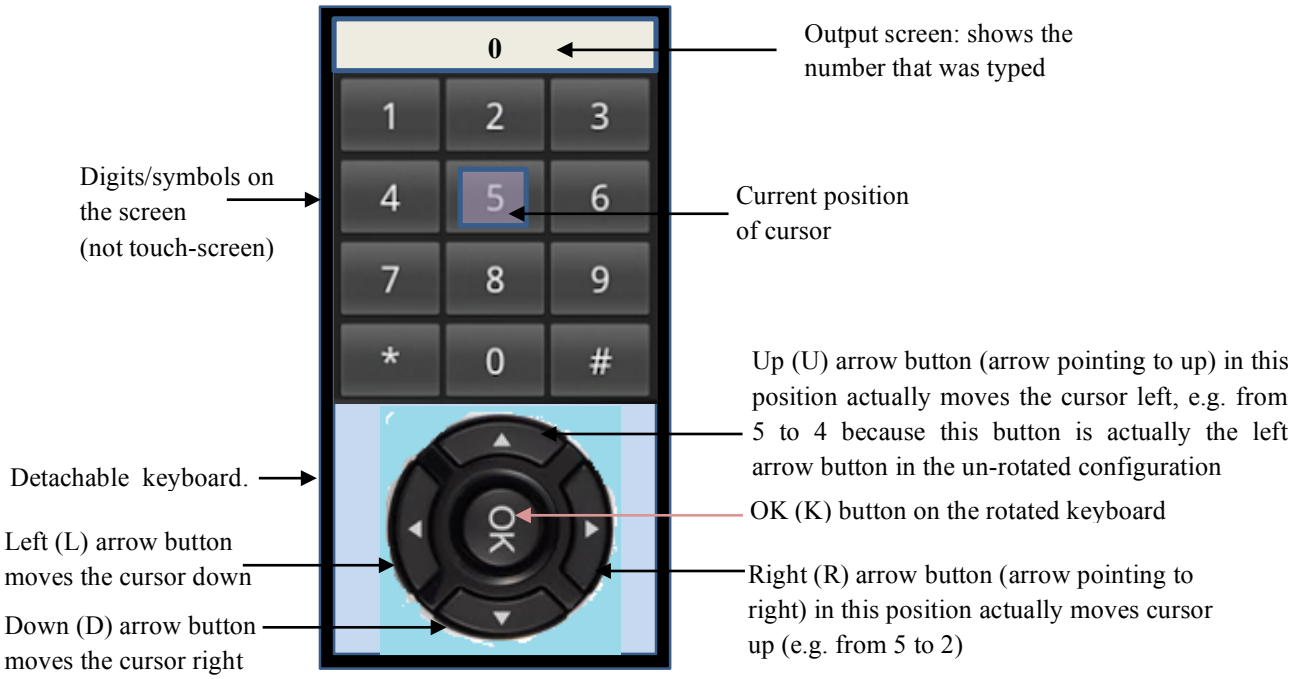

Figure H.2: The cellphone screens and the 90 degree clockwise rotated keyboard. In this case, the Left (L) arrow button corresponding to the observer is the button pointing to Left (i.e., to West). But actually it is the Down button of the un-rotated keyboard. Therefore, pressing the sequence LLK will produce the output 0. Because the first L will move the cursor Down to 8, second L will again move the cursor Down to 0 and K will output 0.

[$]**terms and conditions (read carefully)**: The cursor will *not* move if it is instructed to move to an invalid position (e.g. cursor is at 0 and instructed to move down, it will not move). Also, if the cursor is on the * or # symbol, then it will ignore the OK command, i.e., pressing K will not type the symbol.

Input

The input consists of n (0< n <= 100) cases. Each case starts with two integers r <space> s <space> where r ∈ {0, 90, 180, 270} is the degree of clockwise rotation of the keyboard, and s (<= 10) is the number of scenarios. Then s lines follow, where each line consists of a scenario of the key sequence (max length 50) that was pressed by a user. Assume that each scenario starts with the cursor position at digit 5, and there will be at least one digit output for any scenario.

Output

For each case, you are to output the case number t in one line, then for each scenario, print in one line the number that was typed, without any leading or trailing spaces.

Sample Input / Output

keyboard.in

```
2
0 1
DDKRKRUKUKUK
90 1
DDKRKRUKUKUK
```

OUTPUT

```
1
0963
2
63211
```