Problem 8. Text editor

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	3 seconds
	5 seconds (for Java)
Memory limit:	256 megabytes

When using a simple text editor, the user often needs the reorder of lines in a specific way. If the lines are long, and there is a need to minimize errors, the easiest way to achieve this is by moving chunks of text from one place to another (the "cut/paste" method). You are an efficient programmer. Determine the minimum time necessary to shuffle the lines in the desired way.

The file contains N lines of text. The editor has a single cursor with $N + 1$ possible positions: before the first line of the text, after the last line of the text, and inbetween any two adjacent lines. Pressing "up" or "down" moves the cursor one line up or down, respectively. You cannot move the cursor outside the text.

The Shift key is used to select lines. When it is pressed, the editor remembers the current position of the cursor; when it is released, the editor selects all the lines between the current cursor position and the remembered one. After releasing Shift you must press Ctrl+X to cut the selected lines, i.e. copy them into the clipboard and simultaneously remove them from the text. To paste (insert) back the cut fragment of text from the clipboard, press Ctrl+V after moving the cursor to the desired position using the "up" and "down" keys. After pasting, the text fragment is removed from the clipboard. When a text fragment is cut, all the following lines are automatically moved upward, and when it is pasted, they are moved downward. So there are never any empty lines in the text. When a fragment is cut, the cursor is moved up into the line where the cut fragment began unless it is already there; when a fragment is pasted, the cursor is moved downwards and ends up right after the inserted text.

Every action takes some predefined time, depending on user skills. Initially, the cursor is located before the first line of the text.

Input

In the first line of the input file there is one integer N, being the number of lines in the text editor ($2 \le N \le 8$). In the second line six integers are given, defining how much milliseconds each action takes ($1 \le t_i \le 100$). The list of actions is given below.

N integers in the third line define the initial order of the lines. They are different and lie in the range between 1 and N. N integers in the fourth line define the order in which the lines must be placed in the end. They are also different and lie in the range between 1 and N.

Output

Print two integers into the first line of the output file: T being the total time of executing the plan in milliseconds, and K being the number of actions in the plan. Next, print K actions in the order of their execution, one action per line. Actions are described in the following way:

- `Up` — press "up" (takes t_1 milliseconds).
- `Down` — press "down" (takes t_2 milliseconds).
- `Shift-Press` — press and hold Shift (takes t_3 milliseconds).
- `Shift-Release` — release Shift (takes t_4 milliseconds).
- `Ctrl+X` — press the combination Ctrl+X (takes t_5 milliseconds).
- `Ctrl+V` — press the combination Ctrl+V (takes t_6 milliseconds).

Examples

input.txt	output.txt
6 99 98 100 97 99 98 1 2 3 4 5 6 6 5 4 3 2 1	3252 33 Shift-Press Down Down Shift-Release Ctrl+X Down Down Ctrl+V Shift-Press Down Shift-Release Ctrl+X Down Ctrl+V Shift-Press Up Up Up Shift-Release Ctrl+X Up Up Ctrl+V Down Shift-Press Up Up Up Shift-Release Ctrl+X Up Up Ctrl+V

Example explanation

In the example, you must reverse six lines. Every action takes 100 milliseconds or a bit less. The optimal plan is recorded into an animated gif image, which you can download near these problem statements.