Task 2: Follow me

A *clip* in Live contains musical material, either as a reference to an audio sample or in the form of MIDI notes and controller envelopes. *Follow Actions* allow creating groups of clips that can trigger each other in an orderly or random way (or both). A clip can have two Follow Actions that define what happens after a clip has been playing for a set amount of time.

Every Follow Action has an associated Chance value that controls the likelihood of each of the two Follow Actions occurring. For example, if `action1` has Chance set to 1, and `action2` has Chance set to 0, `action1` will be chosen every time. If, on the other hand, `action2` has Chance set to 10 in this scenario, `action1` will be chosen ten times less often than `action2` on average.

We want you to write a program that simulates this behavior in a simplified way.

Input description

There are two types of input lines; `clip`, which defines a clip with associated Follow Actions, and `ticks`, which drives the playback.

```
clip <name> <ticks_to_play> <follow_chance1> <follow_chance2> <action1> <action2>
```

Each `clip` line defines a clip and its Follow Actions. `ticks_to_play` is the number of ticks that the clip should “play” before it triggers another clip as defined by the Follow Actions. There are two Follow Actions associated with each clip, and they are chosen randomly so that their relative occurrence corresponds to the specified `follow_chances`.

Possible values for `action1` and `action2` are:

- `none` : Nothing happens – the clip keeps playing.
- `any` : Any clip (including the playing one) can be triggered.
- `other` : Any *other* clip can be triggered.
- `next` : The next clip (in the order they were created) will be triggered.
- `previous` : The previous clip will be triggered.

The list of clips should “wrap around”. In other words, if the `next` action is executed on the last clip in the list, the first clip in the list should begin playing. Similarly, executing the `previous` action on the first clip should begin playback of the last clip.

If a clip is created with the same name as an existing clip, the new clip should replace the existing clip.

The second type of input line consists of the word `ticks` followed by the number of ticks to process. The `ticks` line may *optionally* contain a pair of random numbers (between 0 and 1) for *each* tick. (It's part of the task to figure out why two random numbers are needed for each tick.) For example, both of the following `ticks` commands are valid:

```
ticks 3
ticks 3 0.5 0.5 1.0 0.0 0.0 1.0
```

These pairs of random numbers are optionally supplied to ensure that the output can be made deterministic. If these pairs of random numbers are absent in the `ticks` input, real random numbers should be used instead.

With every `ticks` command, the playback should advance the number of ticks given. The very first `ticks` command will begin playback of the first clip.

<u>Output description</u>

For each tick, a single output line is created containing the name of the currently playing clip.

<u>Example</u>

Input

```
clip hello 2 1.0 0.0 next none
clip world 3 1.0 0.0 previous none
ticks 10
```

Output

```
hello
hello
world
world
world
hello
hello
world
world
world
```