Babi likes to see movies every week and he assigns a specific time of each week to watch new movies received during that week. Recently, he has received some new movies in divx format after a long period of time without any new movie! however, all the movies has foreign languages (something except Farsi), so that he tried to download appropriate subtitles of these films. After he downloaded and got disconnected from Internet, he found out that some of the subtitles, which he has downloaded, are not for theare not for the his distribution of the video files. After careful consideration of subtitles, he noticed that the differences may occur in one of the cases below:

- The number of CDs that the subtitle is written for, is different from the number of Babi's video CDs.
- The duration of beginning advertisements in the distribution, which the subtitle is written for, is different from the advertisement duration of Babi's video.
- Some scenes are censored in the Babi's video.

Since Babi prefers to change the current subtitles to match his videos instead of searching for new subtitles, he needs your help to write a program with the following features to manipulate a subtitle:

- Merge n subtitles into one subtitle and/or divide a subtitle into m subtitles.
- Shift a subtitle t seconds backward or forward.
- Cutting a specific period of time from a subtitle.

For those who are not familiar with the format of a regular subtitle file, here is a description of ".srt" format. Every ".srt" subtitle is a text file containing some individual text blocks in the following format:

$BlockIndex$
$StartTime$ `-->` $FinishTime$
$< Sentence\ 1 >$
$< Sentence\ 2 >$
...
$< Sentence\ s_n >$

where $BlockIndex$ shows this block is the $BlockIndex$-th block of the text file, $StartTime$ is the time in which all sentences of this block must be shown and $FinishTime$ is the time the sentences should disappear, all time descriptions are in the format $hh:mm:ss,msmsms$ (e.g. 01:20:01,630 or 99:59:59,999). Each block can have $0 < sn \leq 5$ sentences that should be shown in the specified period. It should be noted that these sentences have at most 80 characters.

All blocks of the movie comes sequentially sorted in increasing order of their start time and the display period of two different blocks are disjoint. Also, the blocks are separated by a blank line. (See the sample input for an example)

Input

Input has several test cases. Each test case starts with the description of Babi's video file. First, the number of CDs, which Babi's video has, $0 < bcdn < 6$, come in a line. Next, $bcdn$ time descriptions come each in a separate line denoting duration of each CD. Next line contains a time description showing the duration of Advertisement *in the beginning of first CD.* After that, a line containing an integer $cn < 100$ come denoting the number of censored parts in the Babi's video, and then, there are cn lines where i-th line contains two time descriptions, which show the begin and the end of the i-th censored part.

Next, the descriptions of subtitle files come, which starts with a single number $0 < scdn < 6$, denoting number of subtitle files. Then, $scdn$ time descriptions each in a separate line come denoting duration of each subtitle file, then a time description comes, which is the duration of advertisement *in the beginning of first CD* of distribution of video, which the subtitle is written for. Finally, the contents of $scdn$ subtitle files come in the above ".srt" format. There is a blank line after each block of test cases.

Note 1. For the simplicity of reading, end of each subtitle file is marked with a Block index 0, but you should know the subtitle files do not contain this 0 at the end, consequently *in the output your program should not print this number at the end of each subtitle file* .

Note 2. There exist at most 1000 blocks in every subtitle file.

Note 3. The time description of censored parts are given *for the distribution of video whose subtitles are downloaded by Babi.*

Note 4. There is no subtitle block in the duration of advertisement.

Note 5. The interval of censored parts are disjoint.

Note 6. All milliseconds between first and last time of a censored part interval *including first and last millisecond* should be deleted from subtitles. For example, a censored part from 00:00:00:001 to 00:00:00:004 cuts *the mathematical interval* $[00{:}00{:}00{:}001, 00{:}00{:}00{:}005)$.

Note 7. Two blocks of subtitles are called *equal* if they have exactly the same sentences. Two blocks of subtitles are called *joint* if the there is not any time between finishing time of one of them and starting time of the other one. *All equal joint blocks must be merged into one block.*

Note 8. All inputs are guaranteed to be valid.

Output

For each test case, your program must output the contents of $bcdn$ subtitles suitable for Babi's video.

Print out a blank line after each of these $bcdn$ subtitle files, and also, after output of each test case.

Sample Input

```
2
00:00:23,000
00:01:00,813
00:00:01,100
2
00:03:35,101 00:03:40,728
00:04:01,000 00:04:10,010
2
00:03:55,813
00:00:28,000
00:03:01,100

1
00:03:14,039 --> 00:03:15,631
Nice car.

2
00:03:19,177 --> 00:03:21,008
Sweet!

3
00:03:30,288 --> 00:03:31,585
Nice.

4
00:03:34,993 --> 00:03:36,893
You little maggot!

5
00:03:38,396 --> 00:03:40,728
-You get back here!
-What are you doing?

6
00:03:40,799 --> 00:03:41,823
This your kid?

7
00:03:41,900 --> 00:03:45,267
Take a look at his hand.
You'll find the valve caps to my '59 Caddy.

8
00:03:45,337 --> 00:03:46,929
Is that true, Jesse?

9
00:03:48,707 --> 00:03:53,041
How many times have I told you
to stop stealing parts off people's cars!

10
00:03:53,845 --> 00:03:55,813
Go on inside. Go!

0

1
00:00:01,630 --> 00:00:04,622
Besides, that was then and this is now.

2
00:00:04,733 --> 00:00:07,167
Yeah, and I should trust you now,
shouldn't l?

3
00:00:07,235 --> 00:00:11,467
Why would I try to steal the golden egg
when I got the whole goose right here?

4
00:00:11,539 --> 00:00:15,202
No. This goose is flying solo
just as soon as we clean this money.

5
00:00:15,777 --> 00:00:18,268
Till then I'm supposed
to automatically trust you?

6
00:00:18,346 --> 00:00:21,042
-Why wouldn't you?
-You're a thief and a bank robber.

7
00:00:21,116 --> 00:00:23,482
I'm not a bank robber, okay?

8
00:00:23,652 --> 00:00:27,247
Look, you're going to get set up
with a little something, all right?

9
00:00:27,322 --> 00:00:28,000
A little something?

0
```

Sample Output

```
1
00:00:14,039 --> 00:00:15,631
Nice car.

2
00:00:19,177 --> 00:00:21,008
Sweet!


1
00:00:07,288 --> 00:00:08,585
Nice.

2
00:00:11,993 --> 00:00:12,100
You little maggot!

3
00:00:12,171 --> 00:00:13,195
This your kid?

4
00:00:13,272 --> 00:00:16,639
Take a look at his hand.
You'll find the valve caps to my '59 Caddy.

5
00:00:16,709 --> 00:00:18,301
Is that true, Jesse?

6
00:00:20,079 --> 00:00:24,413
How many times have I told you
to stop stealing parts off people's cars!

7
00:00:25,217 --> 00:00:27,185
Go on inside. Go!

8
00:00:28,815 --> 00:00:31,807
Besides, that was then and this is now.

9
00:00:31,918 --> 00:00:32,371
Yeah, and I should trust you now,
shouldn't l?

10
00:00:32,372 --> 00:00:33,376
No. This goose is flying solo
just as soon as we clean this money.

11
00:00:33,951 --> 00:00:36,442
Till then I'm supposed
to automatically trust you?

12
00:00:36,520 --> 00:00:39,216
-Why wouldn't you?
-You're a thief and a bank robber.

13
00:00:39,290 --> 00:00:41,656
I'm not a bank robber, okay?

14
00:00:41,826 --> 00:00:45,421
Look, you're going to get set up
with a little something, all right?

15
00:00:45,496 --> 00:00:46,174
A little something?
```