

2003
Gloucester County
High School Programming Contest

Hosted (via the Internet) at Rowan University
Saturday, 15 March 2003

1 Description

Many consumer electronics items, especially home entertainment gear, can be operated by remote control. The remote sends an infrared signal which is interpreted and (if the signal makes sense) causes the device to change its state. It may turn on, or change the volume, or have some other response to the signal.

The remote does not maintain information about the state of the device being controlled, nor does it send out continual signals. When you push ‘Play’, the remote sends out a single signal to the CD player/VCR/&c, just as if you pushed the ‘Play’ button on the control panel.

Your assignment is to write a program which simulates the operation of a particular device. Your program has to report on the device’s changing state as different signals arrive.

1.1 Player States

Our sample machine is a CD player, which will be in one of four activity states: *Play*, *Pause*, *Stop*, and *Sleep*. There is no ‘On’ state, because if the player is on it is either playing, paused, or stopped. The fourth state stands in for ‘Off’, because such machines are never truly off when plugged in (if they were, they wouldn’t be able to interpret the signal from the remote which says to turn on). The player’s start state is always *Sleep*. In addition to what state the player is in, you also have to keep track of what selection is playing and how far that selection has played. To keep matters simple, we will assume that no CD has more than 20 tracks, and no track is more than 30 minutes.

1.2 Player Signals

The signals which can come into the CD player are: **On**, **Sleep**, **Play**, **Pause**, **Stop**, **Ahead**, and **Back**.

On has no effect unless the player is in *Sleep*; when the player is in *Sleep*, no other signal has any effect. The player is always at the beginning of track one in *Stop* when it is first turned on.

Sleep always resets the player to the beginning of track one and puts the player to sleep, regardless of what state it was in before.

Play puts the player in play mode, and it begins advancing through the tracks on the CD, one second advance per one second of time passing. If the player was in *Pause*, **Play** causes it to continue. When the player reaches the end of a track, it continues with the next track. When the player reaches the end of the disc, it switches to *Stop* and resets to the beginning of track one.

Pause has no effect in *Stop* or *Sleep* mode. If the player is in *Play*, **Pause** causes it to stop advancing through the tracks on the disc. In *Pause*, the **Pause** signal causes the player to continue where it left off.

Stop resets the player to the beginning of track one and puts the player in the *Stop* state.

Ahead skips ahead to the beginning of the next track in either *Play* or *Stop*, and the CD player remains in the state it was in. If on the disc’s last track, **Ahead** does nothing at all in any state.

Back sends the player to the beginning of the previous track if the player is in *Stop*, or if the player is in *Play* and its position on the current track is in the first second. If the player is in *Play* partway through a track, **Back** sends the player to the beginning of the track being played. If on the first track, **Back** sends the player back to the beginning of that track.

When in *Pause*, the player cannot skip ahead or back, but can sleep or stop.

While stopped, **Stop** does nothing; while playing, **Play** does nothing.

2 Input/Output Specification

2.1 Input

The input will be in the following format:

1. A single integer, **T**, on a line by itself. This indicates how many tracks are on the disc in the player. This value will be between 1 and 20, inclusive.
2. **T** lines indicating the length of the tracks, each line having only a single positive integer representing duration in seconds. All tracks will be between 1 second and 30 minutes long, inclusive.
3. A single integer, **S**, on a line by itself, indicating how many signals are to be processed. The number of lines to be processed will not be negative.
4. **S** lines indicating the signals being received by the CD player. Each line will have two fields:
 - (a) The time of day that the signal was received, as an eight-character string in the form HH:MM:SS. The time, in 24-hour format, will be between 00:00:00 and 23:59:59, inclusive, and will be zero-padded (*e.g.*, 9:02am would appear as '09:02:00'). Two signals may be less than one second apart, but they will never go backwards.
 - (b) A single integer representing the signal sent as described by this table:

Number	Value	Meaning
1	on	turn the player on
2	sleep	put the player to sleep
3	play	begin playing
4	pause	pause (if playing) or play (if paused)
5	stop	stop the player
6	ahead	skip ahead one track
7	back	skip back one track

There will be no blank lines. You do not need to do error-checking on the input.

Students using a GUI-based system may design an input system that varies from this, provided it maintains the general spirit of this description.

2.2 Output

Program output should be in two sections. The first section should begin with a line saying how many tracks the CD has, and then listing each track number and its duration in MM:SS format. The second section should begin with a line saying how many signals are to be processed, and then one line for each signal. Each line should include the time of day (in 24-hour HH:MM:SS format) the signal arrived, the signal as a word, the state the player was in when the signal arrived (including the track number & position in MM:SS format), and the new state after the signal has been processed (again including the track number & position in MM:SS format). Note that the player's state may change between signals, and your program may have to update the player state based on the signal time before processing the signal itself. Example output:

CD has 2 tracks:

```
1 5:23
2 12:34
```

Processing 3 signals:

```
18:03:22 - ON      was: SLEEP track 1 0:00;    now: STOP track 1 0:00
18:03:23 - PLAY   was: STOP track 1 0:00;    now: PLAY track 1 0:00
18:13:01 - SLEEP  was: PLAY track 2 4:15;    now: SLEEP track 1 0:00
```

Your output does **not** have to conform exactly to the sample output as regards spacing or use of upper/lower case. The samples were formatted to be easy to read, but that is **not** a requirement for your program.

3 Sample Input / Output

3.1 Example One

```
2
323
754
3
18:03:22 1
18:03:23 3
18:13:01 2
```

(The output for this example appears in Section 2.2.)

3.2 Example Two

3.2.1 Input

```
3
60
60
60
10
15:30:00 1
15:30:01 3
15:30:30 4
15:31:50 4
15:33:05 4
15:40:22 3
15:41:00 7
15:41:00 7
15:41:49 6
15:42:26 2
```

3.2.2 Output

CD has 3 tracks:

```
1 1:00
2 1:00
3 1:00
```

Processing 10 signals:

```
15:30:00 - ON      was: SLEEP track 1 0:00;    now: STOP track 1 0:00
15:30:01 - PLAY   was: STOP track 1 0:00;    now: PLAY track 1 0:00
15:30:30 - PAUSE  was: PLAY track 1 0:29;   now: PAUSE track 1 0:29
15:31:50 - PAUSE  was: PAUSE track 1 0:29;  now: PLAY track 1 0:29
15:33:05 - PAUSE  was: PLAY track 2 0:44;   now: PAUSE track 2 0:44
15:40:22 - PLAY   was: PAUSE track 2 0:44;  now: PLAY track 2 0:44
15:41:00 - BACK   was: PLAY track 3 0:22;   now: PLAY track 3 0:00
15:41:00 - BACK   was: PLAY track 3 0:00;   now: PLAY track 2 0:00
15:41:49 - AHEAD  was: PLAY track 2 0:49;   now: PLAY track 3 0:00
15:42:26 - SLEEP  was: PLAY track 3 0:37;   now: SLEEP track 1 0:00
```

4 Test Data

Run your program on this input and make enough screenshots to show the output:

```
11
557
312
270
602
223
160
382
173
213
287
1072
30
08:00:00 1
08:00:01 7
08:00:02 7
08:00:03 3
08:22:01 4
08:27:22 4
08:27:25 7
08:39:25 4
08:47:57 3
08:50:52 2
09:02:02 1
09:02:03 3
09:02:04 6
09:02:05 6
09:02:06 6
09:02:07 6
09:14:41 4
09:18:07 4
09:22:15 6
09:22:16 6
09:22:17 6
09:22:18 6
09:32:30 1
09:32:31 3
09:32:35 4
09:37:37 4
09:37:39 3
09:37:43 7
09:58:22 5
10:02:40 2
```

Your program will be tested on additional data known only to the judges.